IMT School for Advanced Studies, Lucca

Lucca, Italy

Hierarchical Planning and Stochastic Optimization Algorithms with Applications to Self-driving Vehicles and Finance

PHD IN INSTITUTIONS, MARKETS AND TECHNOLOGIES

Curriculum in Control Systems

XXX Cycle

By

Mogens Graf Plessen

2018

Dissertation of Mogens Graf Plessen

Program Coordinator: Prof. Pietro Pietrini, IMT Lucca

Supervisor: Prof. Alberto Bemporad, IMT Lucca

Committee:

Prof. Dario Piga, SUPSI Lugano

Prof. Sergio Savaresi, Politecnico di Milano

IMT School for Advanced Studies, Lucca

2018

Contents

Acknowledgements

I thank Prof. Bemporad for the opportunity to participate in the PhD program at IMT Lucca. I thank all co-authors for any collaboration. Besides Alberto Bemporad, these are Pedro F. Lima, Jonas Mårtensson, Bo Wahlberg, Tommaso Gabriellini, Laura Puglia, Daniele Bernardini and Hasan Esen. Special thanks to Prof. Wahlberg for enabling my visiting stay at KTH Stockholm, Sweden, at the end of 2016, and Pedro F. Lima for the collaboration during that period.

Abstract

This work discusses hierarchical planning and stochastic optimization algorithms with applications to self-driving vehicles and quantitative finance. A diverse set of mathematical tools is considered, ranging from model predictive control (MPC), in both the deterministic and stochastic setting, to various vehicle routing problems, and reinforcement learning using neural networks for function approximation. The applications discussed include single- and multi-automated vehicle motion planning, agricultural in- and out-field logistics planning, as well as dynamic option hedging.

Notation

The number of mathematical symbols which can be used for variables and the like is limited. Therefore, basic notation is here stated for only Section 2.1 which summarizes four publications comprehensively. In contrast, all other sections discuss one publication per section and therefore develop their own notation. Thus, notation has to be interpreseted section-wise. For example, T represents the *transition graph* in the vehicle routing setting of Section 3.1, but also the *option expiration date* in the dynamic hedging setting of Section 4.1.

Main Symbols for Section 2.1

z	State vector
u	Control vector
x	x-coordinate
y	y-coordinate
ψ	Heading angle
t	Time variable
s	Space variable (distance along the road centerline)
$\dot{x}$	Time derivative, $\frac{dx}{dt}$
x'	Spatial derivative, $\frac{dx}{ds}$
$_k$	Subscript to indicate time discretization (e.g., in A_k)
$_j$	Subscript to indicate spatial discretization (e.g., in A_j)
T_s	Sampling time
D_s	Sampling space
v	Velocity
δ	Front-axle steering angle
ρ_s	Radius of curvature
ψ_s	Road centerline heading
e_y	Lateral displacement w.r.t. road centerline
e_ψ	Heading angle w.r.t. road centerline
N	Spatial prediction horizon (e.g., in $j = 0, 1, \ldots, N$)

Main Abbreviations for Section 2.1

MPC	Model Predictive Control
CoG	Center of Gravity
LTI	Linear Time-Invariant
LTV	Linear Time-Varying
LSV	Linear Space-Varying
QP	Quadratic Program
LP	Linear Program
PWA	Piecewise-Affine
SLP	Sequential Linear Program
CPP	Clothoid-based Path Planning
IP	Integer Program
BILP	Binary Integer Linear Program
NMPC	Nonlinear Model Predictive Control

Chapter 1

Introduction

1.1 Four Basic Concepts for Decision Taking

In this thesis, any problem requiring *decision taking* is approached in form of a closed-loop control system, see Figure 1. This section summarizes four basic concepts relevant for closed-loop control system design.

1. Distinction between fast- and slow-sampling systems.

    - According to our definition, a "control system" maps *data* to *decisions*. In practice, this process is repeated. However, it may be repeated often (e.g., in an automotive setting with fast sampling times every 100ms), or less often (e.g., in logistics with slow sampling times once per year or even less frequent).

2. Selection of a) multivariate data sources and b) decision variables.

    - For example, in an automotive setting, cameras, lidars and lasers are frequently used as data sources. Similarly, steering angle and velocity are frequently employed as decision variables. In a financial setting, historical stock prices may serve as the data source. In an agricultural logistics setting, decisions may refer to the assignments of crops to fields.

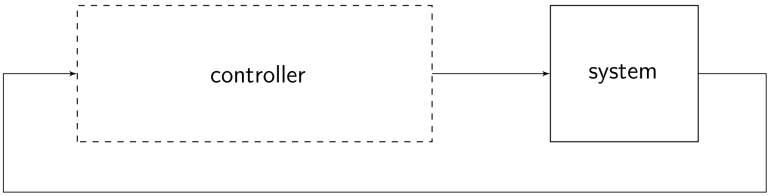
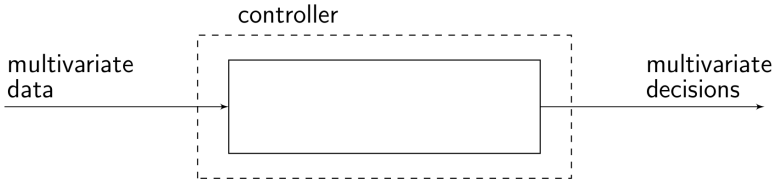
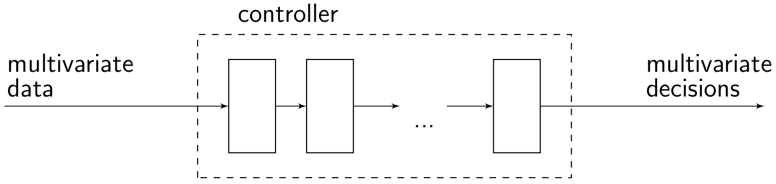


Figure 1: Illustration of a closed-loop control system. A controller maps data to decisions. Then, decisions are applied to a system, before data is measured from the system and again fed to the controller. In practice, this process is repeated. However, dependent on the application, it may be repeated at fast or slower sampling times. In the limit, when the sampling time tends to infinity, the mapping occurs only once.

3. Distinction between a lumped and a hierarchical parametrization.
    - A "lumped controller" implies processing of raw multivariate data. Examples include *end-to-end* controllers mapping raw camera images directly to steering angles.
    - A "hierarchical controller" refers to sequential processing of raw multivariate data. For example, camera images may first be used to extract features such as object locations. Then, in a second hierarchy, these features are processed.
    - See Figure 2 for illustration.
4. Control layer parametrization.
    - A "control layer" denotes the mapping between an input and an output. For lumped controllers, multivariate data is mapped directly to multivariate decision variables. For hierarchical controllers, parametrizations are required for each separate hierarchy. Parametrizations may vary for each hierarchy. In our setting, a state estimator (such as a Kalman Filter) is also interpreted as a potential control layer.
    - Suitably, each control layer is optimized. Therefore, the output of each control layer may result from the solution of an optimization problem. A general *mixed-integer nonlinear nonconvex*



(a) Lumped controller.



(b) Hierarchical controller.

Figure 2: Distinction between (a) lumped and (b) hierarchical controller parametrizations. See Figure 1 for the closed-loop control system.

optimization problem (OP) can be formulated as follows:

$$\inf_{z^{\mathrm{c}}, z^{\mathrm{d}}} \; J(z^{\mathrm{c}}, z^{\mathrm{d}}) \tag{1.1a}$$

$$\text{s.t.} \;\; g_i(z^{\mathrm{c}}, z^{\mathrm{d}}) \leq 0, \quad i = 1, \ldots, m, \tag{1.1b}$$

$$h_i(z^{\mathrm{c}}, z^{\mathrm{d}}) = 0, \quad i = 1, \ldots, p, \tag{1.1c}$$

$$z^{\mathrm{c}} \in \mathbb{R}^{n_{z^{\mathrm{c}}}}, \tag{1.1d}$$

$$z^{\mathrm{d}} \in \mathbb{N}^{n_{z^{\mathrm{d}}}}, \tag{1.1e}$$

where z^{c} and z^{d} summarize continuous and discrete *optimization variables*. Thus, $\mathbb{R}$, $\mathbb{N}$, $\mathbb{R}^n$ and $\mathbb{N}^n$ denote the set of real numbers, the set of integer numbers, the set of real vectors with n scalar elements, and the set of integer vectors with n scalar elements, respectively. The *objective function* is $J : \mathbb{R}^{n_{z^{\mathrm{c}}}} \times \mathbb{N}^{n_{z^{\mathrm{d}}}} \to \mathbb{R}$. *Inequality and equality constraints* are denoted by $g_i : \mathbb{R}^{n_{z^{\mathrm{c}}}} \times \mathbb{N}^{n_{z^{\mathrm{d}}}} \to \mathbb{R}$ and $h_i : \mathbb{R}^{n_{z^{\mathrm{c}}}} \times \mathbb{N}^{n_{z^{\mathrm{d}}}} \to \mathbb{R}$, respectively. The infimum operator "$\inf$" is introduced to generalize for the case that a minimum is not attained. If the problem

is solvable and a minimum exists, $\inf$ can be replaced by the minimum operator "$\min$". Note that (1.1) may be reduced to *quadratic* or *linear programs*. Similarly, a neural network and even an algebraic mapping or a look-up table may be fit. Furthermore, *stochastic* and *deterministic* OPs may be formulated.

- Control layer parametrization comprises the following steps:

  4a OP *formulation*. For example, a linear program (LP) may be used for parametrization.

  4b OP *solution offline* (explicit, or a priori). For example, a LP may be solved offline as a parametric program. Then, during online operation (see Step 4c below), a parametric look-up table can be searched. Another example is the offline computation of a Kalman Filter gain for the design of state estimators. The formulated OP may also not be solved offline at all.

  4c OP *solution online* (implicit, or in real-time). For example, a LP may be solved at every sampling time by a suitable solution algorithm (a solver).

- OP formulations may be adapted to available solution methods or vice versa. For OP formulations, two concepts are of particular relevance: a) exploiting problem structure for *parallelization*, and b) *preconditioning* (problem transformations and scalings) that alleviate OP solutions.

To summarize, in this thesis, problems that involve decision taking are approached by the application of above four concepts. First, the closed-loop control system sampling time is decided upon. Second, data and decision variables are determined. Third, a control system architecture with at least one control layer is selected. Finally, each control layer is parameterized, whereby it must be decided on a) an OP formulation, b) an OP solution offline (i.e., in form of a preparatory step), and c) an OP solution online (i.e., for real-time operation).

1.2 The Individual and Societal Aspect for the Automation of Transportation

Within the context of automated driving, it is distinguished between optimization of the *individual benefit* and the *societal benefit*.

The Individual Aspect

From an individual's perspective, desirable transportation means are characterized by being: safe, reliable, environmental friendly, fast and affordable. It can be distinguished between two hierarchical planning layers. These are a) the high-level *vehicle routing* layer, and b) the low-level *vehicle motion* layer. The former is for high-level route planning, the latter for low-level plan execution. Therefore, there are two technological demands for a) a method for the motion control of an agile agent (the self-driving vehicle) in its surrounding, and b) for a method that devises routes (missions) for the agile agent under consideration of the present surrounding in which it interacts.

The Societal Aspect

Rather than perceiving the surrounding as a given (uncontrollable) "constraint", the individual benefit can be further improved when jointly also optimizing the surrounding. In the transportation setting, the surrounding refers to a) other vehicles, and b) the infrastructure. As a consequence, the following classification can be made:

1. Single- and multi-vehicle motion planning (local level).
2. Single- and multi-vehicle routing (global level).
3. Infrastructure control.

An example for multi-vehicle motion planning on a local level is the control of multi-automated vehicles within a local road segment. In contrast, multi-vehicle routing implies mission planning within a larger area, for example, within a city. Infrastructure control mainly refers to traffic

light control, at intersections or along highways. Finally, note that in all cases, motion control of each agile agent remains the fundamental prerequiste. Each agent may be fully automated, semi-automated or manually controlled.

The topics covered in this thesis for the automation of transportation are summarized as follows. In Chapter 2 full automation is targeted, and methods for single and multi-vehicle motion planning are presented. In contrast, in Chapter 3 methods for vehicle routing are discussed.

1.3 Continuous- and Discrete-time System Modeling

All control methods presented in this thesis leverage mathematical system *models*. Either for model-based control and optimization, or for model-based training of control parametrizations in a reinforcement learning setting. Because of their importance, basic system model representations are discussed in the following.

Nonlinear Continuous-time Dynamic System Modeling

In this thesis, nonlinear continuous-time dynamic systems are modeled as first-order *ordinary differential equations* (ODEs), $\dot{z}(t) = f(z(t), u(t), t)$, or abbreviated, $\dot{z} = f(z, u, t)$, and further abbreviated,

$$\dot{z} = f(z, u), \tag{1.2}$$

with $t \in \mathbb{R}$ indicating time, derivative $\dot{z} = \frac{dz}{dt}$, and where $z \in \mathbb{R}^{n_z}$ and $u \in \mathbb{R}^{n_u}$ denote *state* and *input* (decision) vectors, respectively. Note that a system such as $\tilde{z}^{(n)} + \tilde{f}(\tilde{z}, \dot{\tilde{z}}, \ddot{\tilde{z}}, \ldots, \tilde{z}^{(n-1)}, u) = 0$ can be brought to the form of (1.2) by the coordinate transformation $z = \begin{bmatrix} \tilde{z} & \dot{\tilde{z}} & \ddot{\tilde{z}} & \ldots & \tilde{z}^{(n-1)} \end{bmatrix}$.

Nonlinear Discrete-time Dynamic System Modeling

There exist a variety of methods to discretize (1.2), see [71]. The simplest one is the *Euler forward* scheme with

$$z_{k+1} = z_k + h f(z_k, u_k) \quad \text{and} \quad t_{k+1} = t_k + h, \tag{1.3}$$

where h denotes the discretization step size and z_{k+1} approximates $z(t_{k+1})$. Typically, h is selected as the sampling time T_s such that $h = T_s$. An alternative more accurate method is the *fourth-order Runge-Kutta* scheme

$$z_{k+1} = z_k + \frac{h}{6}(c_1 + 2c_2 + 2c_3 + c_4), \tag{1.4a}$$

$$c_1 = f(z_k, u_k, t_k), \tag{1.4b}$$

$$c_2 = f(z_k + h\frac{c_1}{2}, u_k, t_k + \frac{h}{2}), \tag{1.4c}$$

$$c_3 = f(z_k + h\frac{c_2}{2}, u_k, t_k + \frac{h}{2}), \tag{1.4d}$$

$$c_4 = f(z_k + hc_3, u_k, t_k + h), \tag{1.4e}$$

with $t_{k+1} = t_k + h$, and assuming $u(t) = u_k,\ \forall t \in [t_k, t_{k+1}]$.

Linear Time-Invariant Continuous-Time Dynamic System Modeling

Using a *first-order Taylor approximation*, the linearization of (1.2) is[1]

$$\dot{z} = f(z^{\text{ref}}, u^{\text{ref}}) + A^c(z - z^{\text{ref}}) + B^c(u - u^{\text{ref}}), \tag{1.5}$$

with $A^c = \frac{\partial f(z^{\text{ref}}, u^{\text{ref}})}{\partial z}$ and $B^c = \frac{\partial f(z^{\text{ref}}, u^{\text{ref}})}{\partial u}$ denoting the partial derivatives with respect to z and u, and evaluated at *references* z^{ref} and u^{ref}.

Linear Time-Invariant Discrete-Time Dynamic System Modeling

Under the assumption of $u(t) = u_k$, and $k \in \mathbb{N}$ indexing steps over $t \in [kT_s, (k+1)T_s]$ with sampling time T_S, the *exact discretization* of (1.5) is obtained as

$$\begin{aligned} z_{k+1} &= e^{A^c T_s} z_k + \left(\int_0^{T_s} e^{A^c \eta} d\eta \right) \left(B^c(u_k - u^{\text{ref}}) + f(z^{\text{ref}}, u^{\text{ref}}) - A^c z^{\text{ref}} \right), \\ &= Az_k + Bu_k + g, \end{aligned} \tag{1.6}$$

where (integrated) matrix exponentials can be evaluated according to [361], and where A, B and g summarize different contributions. Note

[1]For simplicity, equality-signs "=" instead of approximation-signs "≈" are used throughout this section. This is to be considered when comparing (1.5) and (1.2).

that $e^{A^c T_s} = \sum_{m=0}^{\infty} \frac{(A^c T_s)^m}{m!}$. Thus, (1.6) represents a *linear time-invariant* (LTI) discrete-time dynamic system model.

Linear Time-Varying Continuous-Time Dynamic System Modeling

Rather than assuming time-invariant references z^{ref} and u^{ref} in (1.5), the *time-varying* correspondent can be formulated as

$$\dot{z} = f(z_k^{\text{ref}}, u_k^{\text{ref}}) + A_k^c(z - z_k^{\text{ref}}) + B_k^c(u - u_k^{\text{ref}}), \text{ for } t \in [kT_s, (k+1)T_s], \quad (1.7)$$

where z_k^{ref} and u_k^{ref} denote references constant over an interval $t \in [kT_s, (k+1)T_s]$, but in general time-varying over different intervals indexed by $k \in \mathbb{N}$.

Linear Time-Varying Discrete-Time Dynamic System Modeling

For (1.7), the *linear time-varying* (LTV) discrete-time correspondent to (1.6) is then derived as

$$z_{k+1} = A_k z_k + B_k u_k + g_k, \quad (1.8)$$

with state transition matrix A_k as well as B_k constant over $t \in [kT_s, (k+1)T_s]$, but varying over $k \in \mathbb{N}$.

Integrating Logical Constraints

For the formulation of logistical optimization problems, three classes of *logical constraints* are of particular interest. They are translated into integer linear inequalities using [372], [40]. Let $\epsilon > 0$ be a small number (e.g., the machine precision), $b, b_1, b_2, b_3 \in \{0, 1\}$, $y \in \mathbb{R}$, and $g(z)$ such that $g : \mathbb{R}^{n_z} \to \mathbb{R}$ is linear, n_z the variable dimension, $g^{\max} = \max_{z \in \mathcal{Z}} g(z)$ and $g^{\min} = \min_{z \in \mathcal{Z}} g(z)$, and where $\mathcal{Z}$ is a given bounded set.

1. The statement "$b = 1$ if and only if $g(z) \leq 0$ and $b = 0$ otherwise" is equivalent to

$$g(z) \leq g^{\max}(1 - b), \quad g(z) \geq \epsilon + (g^{\min} - \epsilon)b. \quad (1.9)$$

2. The statement "$b_3 = 1$ if and only if $b_1 = 1$ and $b_2 = 1$, and $b_3 = 0$ otherwise" is equivalent to $b_3 = b_1 b_2$ and is equivalent to

$$b_1 + b_2 - b_3 \leq 1, \quad b_3 \leq b_1, \quad b_3 \leq b_2. \tag{1.10}$$

3. The statement "$y = g(z)$ if $b = 1$ and $y = 0$ otherwise" is equivalent to $y = bg(z)$ and is equivalent to

$$\begin{aligned} &y \leq g^{\max} b, \quad y \geq g^{\min} b, \quad y \leq g(x) - g^{\min}(1 - b), \\ &y \geq g(x) - g^{\max}(1 - b). \end{aligned} \tag{1.11}$$

1.4 Motivation of this Thesis

In general, two very complex and multi-faceted problem classes are *self-driving vehicles* and *quantitative finance*. This motivated to select these as testbeds for the application and development of hierarchical planning and stochastic optimization algorithms. Both problem classes are approached using the aforementioned four basic concepts.

While special focus is on *model predictive control* (MPC) [269], [65] as the general control concept (in both the *deterministic* and *stochastic* setting), a variety of alternative techniques are also employed for the parametrization of at least some of hierarchical control layers. Therefore, a second motivation for this thesis is an empirical analysis of the advantages and the deficiencies of MPC in comparison to alternatives for the two problem classes considered.

1.5 Thesis Outline and List of Publications

This section summarizes the contribution by listing publications developed between November 4, 2014 and December 4, 2017. The chapters in which these publications are discussed are also indicated. The final version of this thesis was completed on January 14, 2018.

CHAPTER 2:

VEHICLE MOTION PLANNING

1. **M. Graf Plessen**, "Automating vehicles by deep reinforcement learning using task separation with hill climbing," *arXiv preprint arXiv: 1711.10785*, 2017. (Submitted).

2. **M. Graf Plessen**, "Trajectory planning of automated vehicles in tube-like road segments," in *IEEE Conference on Intelligent Transportation Systems*, pp. 83-88, 2017.

3. **M. Graf Plessen**, P.F. Lima, J. Mårtensson, A. Bemporad, and B. Wahlberg, "Trajectory planning under vehicle dimension constraints using sequential linear programming," in *IEEE Conference on Intelligent Transportation Systems*, pp. 108-113, 2017.

4. **M. Graf Plessen**, D. Bernardini, H. Esen, and A. Bemporad, "Spatial-based predictive control and geometric corridor planning for adaptive cruise control coupled with obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 26(1), pp. 38-50, 2018.

5. **M. Graf Plessen**, and A. Bemporad, "Reference trajectory planning under constraints and path tracking using linear time-varying model predictive control for agricultural machines," *Biosystems Engineering*, vol. 153, pp. 28-41, 2017.

6. **M. Graf Plessen**, D. Bernardini, H. Esen, and A. Bemporad, "Multi-automated vehicle coordination using decoupled prioritized path planning for multi-lane one- and bi-directional traffic flow control," in *IEEE Conference on Decision and Control*, pp. 1582-1588, 2016.

CHAPTER 3:

VEHICLE ROUTING

1. **M. Graf Plessen**, "Path planning for area coverage," *WO Patent App. PCT/EP2016/072966*, June 8 2017.

2. **M. Graf Plessen**, "System and method for navigation guidance of a vehicle in an agricultural field," *WO Patent App. PCT/EP2016/072968*, June 8 2017.

3. **M. Graf Plessen**, "Partial field coverage based on two path planning patterns," draft at *arXiv preprint arXiv: 1707.07649*, 2017.

4. **M. Graf Plessen**, "Coupling of crop assignment and vehicle routing for harvest planning in agriculture," *arXiv preprint arXiv: 1703.08999*, 2017. (Submitted).

5. **M. Graf Plessen**, "Coordination of harvesting and transport units for area coverage," *PCT/IB2017/051899*, June 4 2017.

6. **M. Graf Plessen**, and A. Bemporad, "Shortest path computations under trajectory constraints for ground vehicles within agricultural fields," in *IEEE Conference on Intelligent Transportation Systems*, pp. 1733-1738, 2016.

CHAPTER 4:

DYNAMINC OPTION HEDGING AND STOCK TRADING

1. **M. Graf Plessen**, L. Puglia, T. Gabbriellini, and A. Bemporad, "Dynamic option hedging with transaction costs: A stochastic model predictive control approach," *International Journal of Robust and Nonlinear Control*, pp. 1-20, 2017.

2. **M. Graf Plessen**, and A. Bemporad, "Parallel investments in multiple call and put options for the tracking of desired profit profiles," in *IEEE American Control Conference*, pp. 1091-1096, 2017.

3. **M. Graf Plessen**, and A. Bemporad, "A posterior multi-stage optimal trading under transaction costs and a diversification constraint," *arXiv preprint arXiv: 1709.07527*, 2017. (Submitted).

4. **M. Graf Plessen**, and A. Bemporad, "Stock trading via feedback control: Stochastic model predictive or genetic?," originally presented as a poster at *XVIII Workshop on Quantitative Finance (QFW2017)*, to appear upon invitation in *Journal of Modern Accounting and Auditing*, available at *arXiv preprint arXiv: 1708.08857*, 2017.

SYSTEM IDENTIFICATION

The following three contributions are also listed since their final versions in revised form were completed at the beginning of above mentioned time span. Since they are the result of the master's thesis they are otherwise not further discussed.

1. **M. Graf Plessen**, T.A. Wood, R.S. Smith, "Nuclear norm minimization algorithms ofr subspace identification from non-uniformly spaced frequency data," in *IEEE European Control Conference*, pp. 2032-2037, 2015.

2. **M. Graf Plessen**, V. Semeraro, T.A. Wood, R.S. Smith, "Optimization algorithms for nuclear norm based subspace identification with uniformly spaced frequency domain data," in *IEEE American Control Conference*, pp. 1119-1124, 2015.

3. **M. Graf Plessen**, T.A. Wood, R.S. Smith, "Time-domain subspace identification algorithms using nuclear norm minimisation," in *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 903-908, 2015.

Chapter 2

Single and Multi-Vehicle Motion Planning

Self-driving vehicles (terrestrial, aerial and marine) are perceived to be the future of transportation and an enabling technology for the transformation of logistics. One promising application is the relief of humans from operating in rough, repetitive or exhausting environments. Examples include mining, agriculture, urban robot-taxis, and the automation of goods and freight transport through relentless 24/7/365 operation.

Automated vehicles can address various challenges. Fuel consumption can be reduced by means of platooning [5], and anticipative driving in car-2-car and car-2-infrastructure communicating environments [186], [141]. Traffic safety may be increased by means of automated handling of vehicles at their friction limits [268], [8], [120]. Congestion in cities can be reduced by means of coordinated traffic flows [192]. We can distinguish between longitudinal and steering-related vehicle control. The former is much simpler when considered isolatedly and it is introduced commercially [52]. Steering-applications are more complicated, since the exact traveling trajectory is decisive for permissible traveling speeds within friction limits, thereby affecting vehicle stability. In general, we can distinguish between high- and low-velocity driving scenarios. For the former, steering is relevant for obstacle avoidance and throughput maxi-

mization on highways with vehicles of different agility capabilities [154]. For the latter, steering is relevant for tight maneuvering.

In this chapter it is outlined how different aspects that are typical for automated vehicle motion planning are addressed. Therefore, a) single-vehicle motion planning by MPC, b) single-vehicle motion planning by neural networks, and c) multi-vehicle motion planning are discussed. Sometimes, the preferred solution out of two options or an emphasized aspect are highlighted in a **SUMMARY**.

2.1 Single-Vehicle Motion Planning by MPC

This section summarizes the publications [148], [156], [155] and [152]:

- M. Graf Plessen, "Trajectory planning of automated vehicles in tube-like road segments," in *IEEE Conference on Intelligent Transportation Systems*, pp. 83-88, 2017.

- M. Graf Plessen, P.F. Lima, J. Mårtensson, A. Bemporad, and B. Wahlberg, "Trajectory planning under vehicle dimension constraints using sequential linear programming," in *IEEE Conference on Intelligent Transportation Systems*, pp. 108-113, 2017.

- M. Graf Plessen, D. Bernardini, H. Esen, and A. Bemporad, "Spatial-based predictive control and geometric corridor planning for adaptive cruise control coupled with obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 26(1), pp. 38-50, 2018.

- M. Graf Plessen, and A. Bemporad, "Reference trajectory planning under constraints and path tracking using linear time-varying model predictive control for agricultural machines," *Biosystems Engineering*, vol. 153, pp. 28-41, 2017.

This secion focuses on motion planning by MPC. For MPC, the conventional and typical control layer parametrization is a *quadratic program* (QP) or a *linear program* (LP). Here, this convention is followed. Thus, for MPC, only QPs or LPs are considered as candidate parametrizations.

2.1.1 LTI- or LTV-MPC formulation

The first question to address is whether to use a *linear time-invariant* (LTI) or *linear time-varying* (LTV) MPC formulation for motion planning of automated vehicles by MPC. Therefore, to answer this question, the simple nonlinear kinematic bicycle model [317],

$$\begin{bmatrix} \dot{x} & \dot{y} & \dot{\psi} \end{bmatrix}^\mathsf{T} = \begin{bmatrix} v\cos(\psi) & v\sin(\psi) & \frac{v}{l}\tan(\delta) \end{bmatrix}^\mathsf{T}, \tag{2.1}$$

is assumed as vehicle model. See Figure 3 for notation. Typically, the center of gravity (CoG) is located at the rear axle and l denotes the wheelbase. The front-axle steering angle δ and vehicle velocity v are used as control variables. The following linearly constrained optimization problem with quadratic objective function (QP) is formulated:

$$\min \ \sum_{k=1}^{K} \alpha_x(x_k - x_k^{\text{ref}})^2 + \alpha_y(y_k - y_k^{\text{ref}})^2 + \alpha_\psi(\psi_k - \psi_k^{\text{ref}})^2 \tag{2.2a}$$

$$\text{s.t.} \ \ z_0 = z(0), \tag{2.2b}$$

$$z_{k+1} = \begin{cases} Az_k + Bu_k + g, & k = 0, \dots, K-1, \text{ LTI-MPC} \\ A_k z_k + B_k u_k + g_k, & k = 0, \dots, K-1, \text{ LTV-MPC} \end{cases} \tag{2.2c}$$

$$u^{\min} \leq u_k \leq u^{\max}, \ k = 0, \dots, K-1, \tag{2.2d}$$

with optimization variables $\{u_k\}_{k=0}^{K-1}$, $u = \begin{bmatrix} v & \delta \end{bmatrix}^\mathsf{T}$, optimization horizon K, subscript k indexing sampling times over the optimization horizon, $z = \begin{bmatrix} x & y & \psi \end{bmatrix}^\mathsf{T}$, $z(0)$ the current state at the instant of planning, and (2.2c) indicating the linearized and discretized vehicle dynamics of (2.1) for LTI- and LTV-MPC according to Section 1.3, respectively. Rate constraints are omitted to fully focus on the effect of LTI- and LTV-formulation. Two experiments are conducted, whereby a refence trajectory is tracked. The reference path is generated with a spatial discretization of $D_s = 0.5$m. Remaining parameters are set as follows: $T_s = 0.25$s, $u^{\max} = \begin{bmatrix} 50/3.6 & 40\frac{\pi}{180} \end{bmatrix}$, $u^{\min} = \begin{bmatrix} 5/3.6 & -40\frac{\pi}{180} \end{bmatrix}$, $u_k^{\text{ref}} = \begin{bmatrix} D_s/T_s & 0 \end{bmatrix}^\mathsf{T}$, $\forall k$, and $\alpha_x, \alpha_y, \alpha_\psi = 1$. All parameters are in SI-units. For LTI-MPC, $z(0)$ and u_0^{ref} are used for linearization and discretization. The experimental results are visualized in Figures 4 and 5. Results are similar for a reduced

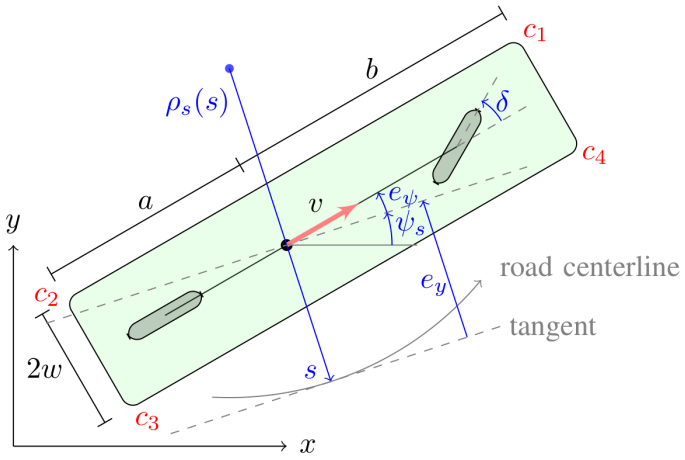


Figure 3: A bicycle model, including the representation of the curvilinear (road-aligned) coordinate system, and vehicle dimensions.

sampling time of $T_s = 0.1$s. For all four experiments, (2.2) is solved once for a vehicle start position located at the origin, i.e., without conducting a receding horizon simulation. These four experiments are sufficient to formulate the following fundamental observation:

SUMMARY 1. *LTI-system modling is unsuitble for general motion planning of self-driving vehicles. The nonholonomic nature of vehicles and the corresponding system nonlinearities are not sufficiently well approximated by a LTI system model.*

As further outlined in Section 2.1.5, in this thesis it is distinguished between *kinematic* and *dynamic vehicle models*. Since the latter models are more complicated than the former through the inclusion of additional tire models, Summary 1 also applies for dynamic vehicle models.

Summary 1 has important implications. Foremostly, the entire machinery for finite horizon *linear* MPC [65], mainly with regard of stability, for example, via *terminal state constraint sets* invariant for *terminal controllers* [270], cannot be applied [104]. Summary 1 also precludes the use of *explicit MPC* [41] for general motion planning.

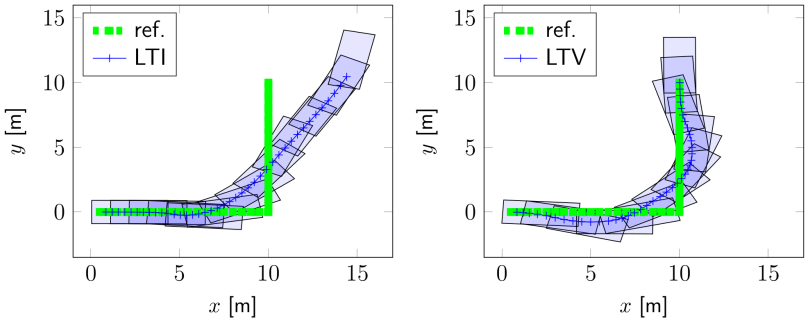


Figure 4: Example 1 of Section 2.1.1. Comparison of the solution of (2.2) for the LTI (left) and (LTV) case when tracking the green reference, a $90°$-turn (an ubiquitous driving scenario). Vehicle dimensions are displayed every third discretization index. For both cases, $K = 40$.

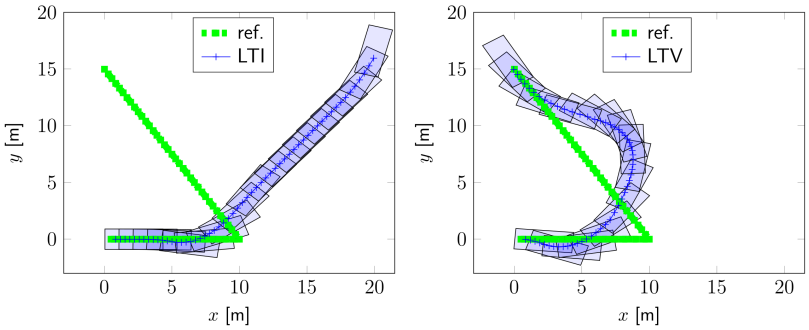


Figure 5: Example 2 of Section 2.1.1. Comparison of the solution of (2.2) for the LTI (left) and (LTV) case when tracking the green reference. Vehicle dimensions are displayed every third discretization index. For both cases, $K = 56$.

Extending Remarks

Remark 1. *For nonlinear system model* (2.1)*, exact discretization can be derived analytically under the assumption of constant* $u(t) = u_k, \forall t \in [kT_s, (k+1)T_s]$ *and* $k \in \mathbb{N}$.

Proof. For dynamics (2.1), the continuous-time state transition matrix after linearization is *nilpotent*. Therefore, the Taylor series representing the matrix exponentials in (1.6) is analytical. See [152] for details. □

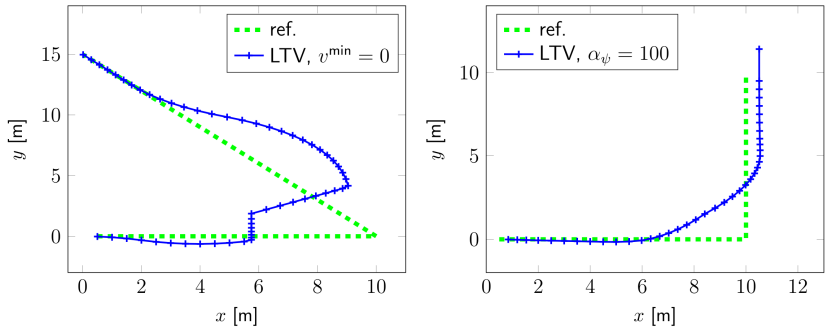


Figure 6: Example 3 and 4 of of Section 2.1.1. Visualization of the solution of (2.2) (solved once when planning at $z(0) = [0\ 0\ 0]^\mathsf{T}$) for the LTV-case when tracking the green reference. Experimental setup is identical to the one for Experiment 1 and 2, except that $v^{\min}$ and α_ψ are adjusted, respectively.

Remark 1 is relevant for an embedded LTV-MPC controller implementation. It encourages to design motion planners based on nonlinear *kinematic* bicycle models since linearized and discretized dynamics can be computed analytically and accordingly fast for online MPC.

Based on above findings, two further experiments are reported. The focus is on the LTV-MPC formulation. The results are displayed in Figure 6. Several remarks can be made. First, while actuator absolute (and in general also rate) constraints are physically prescribed, their artificial bounding can have a notable influence. As illustrated, decreasing $v^{\min} = 5/3.6$ to $v^{\min} = 0$ has a significant effect on the smoothness of the trajectory from the solution of (2.2). The resulting "optimized" control signal v_k is very small for part of the prediction horizon. Second, a modification of the objective function weight can significantly influence the tracking result. Note that both examples are stated only to *illustrate* the sensitivity of MPC results to MPC formulations. The steady state tracking error in Example 4 may, for example, be avoided by a state-space extension for integral action.

SUMMARY 2. *The solution trajectories output by MPC are often strongly affected by the optimization problem formulation, which includes objective function as well as state and control constraints design.*

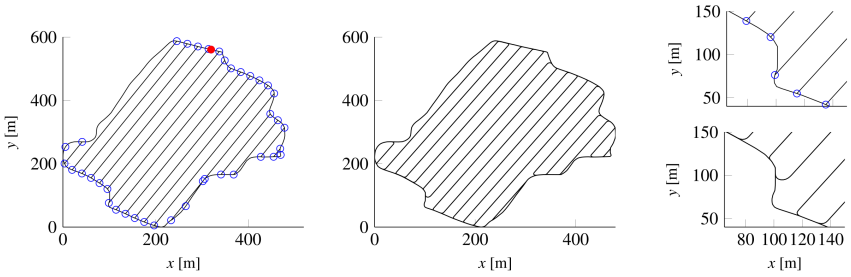
2.1.2 Case Study: The Influence of Reference Trajectories

To further underline the importance and influence of reference trajectories, the case study from [152] is discussed. Therefore, offline reference trajectory generation tailored for high- precision closed-loop tracking in agricultural fields is considered.

It is assumed that an in-field logistical optimization step has determined an *edgy* field coverage path. Then, this section limits the discussion of [152] to the *smoothing* of this edgy path under consideration of various constraints such that the resulting reference trajectory admits high-precision tracking by an autonomous ground vehicle using LTV-MPC, see Figure 7.

Three Design Trajectories for Path Smoothing

Circle-segments, *generalized elementary paths* and *bi-elementary paths* are considered as design elements for path smoothing. A *circle* trajectory is characterized by having an instantaneous centre of rotation (CoR) and constant curvature $\mathcal{C}$, i.e., $\mathcal{C}(s) = 1/R$, where R is the circle radius and $s \in [0, L]$ with L the path length of the circle-segment. A *generalized elementary path* is presented as a tool for our application. The work of [121], which uses two concatenated generalized elementary paths for emergency lane change trajectories when operating autonomous passenger vehicles at their friction limits, is summarized. The theoretical basis was developed by [207] and [328]. Let there be two coordinates $P_i = [x_i, y_i]^\mathsf{T}$, $i = 1, 2$, which we wish to connect. We arbitrarily select the initial heading direction of P_1 as $\psi_1 = 0$, see Figure 8. By translation and rotation any other orientation can be achieved. Parameter $\lambda \in [0, 1)$ determines the arc fraction length. A circle-segment is described by $\lambda \to 1$. A purely clothoid-based trajectory is implied by $\lambda = 0$. For $0 < \lambda < 1$, a generalized elementary path consists of entry clothoid, arc and exit clothoid. Equations describing positions, $x(s)$ and $y(s)$, and heading, $\psi(s)$, along



(a) Start constellation (b) Desired: smoothed path (c) Close-ups

Figure 7: An illustrative real-world field. (a) Starting constellation: edgy tractor lanes, nodes (blue circles) and entry/exit of field (red dot). An edgy path plan for field coverage is described as a sequence of node visits that is obtained from an in-field logistical optimization step [146]. (b) A *smoothed* reference trajectory tailored for high-precision tracking by an autonomous tractor-system. (c) Close-ups for detailed visualization.

path coordinate $s \in [0, L]$ are derived as

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \tag{2.3a}$$

$$\alpha = \psi_2 - \psi_1 = 2\tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right), \tag{2.3b}$$

$$D = 2\int_0^{\frac{\lambda}{2}} \cos\left(\frac{2\alpha}{1+\lambda}\eta\right) d\eta + \ldots, \tag{2.3c}$$

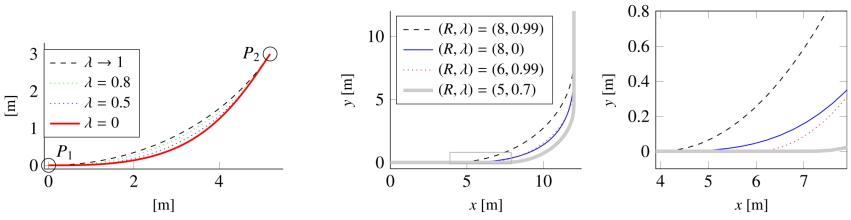
$$+ 2\int_{\frac{\lambda}{2}}^{\frac{1}{2}} \cos\left(\frac{2\alpha}{1-\lambda^2}\left(-\eta^2 + \eta - \frac{\lambda^2}{4}\right)\eta\right) d\eta, \tag{2.3d}$$

$$L = \frac{d}{D},\ \sigma = \frac{4\alpha}{L^2(1-\lambda^2)}, \tag{2.3e}$$

$$\psi(s) = \begin{cases} \psi_1(s) = \int_0^s \sigma\eta d\eta,\ s \in [0, L\frac{1-\lambda}{2}] \\ \psi_2(s) = \int_{L\frac{1-\lambda}{2}}^s \sigma L\frac{1-\lambda}{2} d\eta + \psi_1(L\frac{1-\lambda}{\eta}),\ s \in (L\frac{1-\lambda}{2}, L\frac{1+\lambda}{2}) \\ \psi_3(s) = \int_{L\frac{1+\lambda}{2}}^s \sigma(L-\eta) d\eta + \psi_2(L\frac{1+\lambda}{2}),\ s \in [L\frac{1+\lambda}{2}, L] \end{cases} \tag{2.3f}$$

$$x(s) = \int_0^s \cos(\psi(\eta)) d\eta + x_1,\ s \in [0, L], \tag{2.3g}$$

$$y(s) = \int_0^s \sin(\psi(\eta)) d\eta + y_1,\ s \in [0, L]. \tag{2.3h}$$



(a) Generalised elementary path.

(b) 90°-transition.

(c) Close-up.

Figure 8: (a) Influence of $\lambda \in [0, 1)$ on the shape of a *generalized elementary path*. A circle-segment is given by $\lambda \to 1$. (b) A 90°-transition for four different parameter selections. (c) Close-up for detailed visualization. The close-up emphasizes the influence of parameter selections on start of turning. While the case $(R, \lambda) = (6, 0.99)$ exceeds $y = 0$ only beyond the $x = 6$m-mark, the clothoid-based design $(R, \lambda) = (8, 0)$ does so before the 5m-mark.

Analytical solutions to (2.3d) and part of (2.3f) do not exist. Therefore, a simple *midpoint rule* for numerical integration is employed. A circle-segment can be replaced with a generalized elementary path connecting the two points, P_1 and P_2, while maintaining the same heading directions at both P_1 and P_2.

Our third design trajectory is a *bi-elementary path*, which is a concatenation of two generalized elementary paths and parameterized by the *symmetric point fraction* γ. For brevity of illustration, it is here not further discussed. See [152] for details.

To summarize, to generate a smooth transition between two locations (e.g., to model a 90° turn), it remains to choose one of the three design trajectories and to select parameter λ, and potentially γ.

Remark 2. *The path primitives straights, arcs and clothoids can also be concatenated for clothoid-based path planning (CPP). Interestingly, according to [139], path planning methods for automated vehicles most applied in real implementations by research groups worldwide are interpolation-based. Clothoids, together with Bézier and polynomial curves, belong to that class. Corners of safety margin-adjusted obstacles may serve as waypoints. See [156] for a detailed discussion of an experiment and a comparison to MPC in a tight maneuvering space. The main benefit of MPC-based approaches over CPP methods*

is that MPC more easily permits anticipative steering accounting for the entire obstacle configuration. In contrast, CPP methods largely rely on the selection of waypoints, which are then connected by path primitives.

Influence of System Constraints

The influence of system constraints on a smooth path trajectory is discussed; considering l, $\delta^{\max}$, $\dot{\delta}^{\max}$, T_s, $\bar{v}^{\text{ref}}$ and the selection of one of the design trajectories. The maximum steering angle and rate of the vehicle are $\delta^{\max}$ and $\dot{\delta}^{\max}$. The control system sampling time is T_s. Trajectories (curves) are traversed at constant speed $\bar{v}^{\text{ref}}$. In the following, means for determining lower bounds on turning radius R are described.

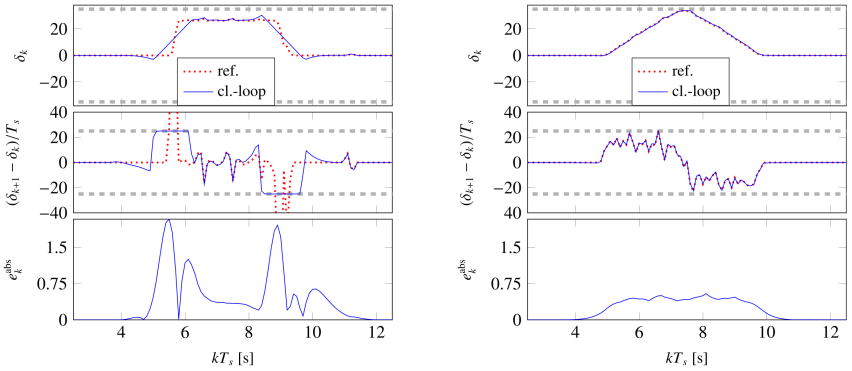
Considering the nominal model, (2.1), let time index $k \in \mathbb{Z}_+$ be associated with sampling time T_s such that all times of interest can be described as kT_s. Assuming constant input signals, we obtain by integration $\psi_{k+1} = \frac{\bar{v}^{\text{ref}} T_s}{l} \tan(\delta_k) + \psi_k$, with the abbreviation $\psi_k = \psi(kT_s)$. The constraints of interest are $|\delta_k| \leq \delta^{\max}$ and $|\delta_{k+1} - \delta_k| \leq \dot{\delta}^{\max} T_s, \forall t \in [k_1 T_s, k_2 T_s]$, whereby k_1 and k_2 define the time interval for curve traversal. Thus,

$$\delta_k = \tan^{-1}\left((\psi_{k+1} - \psi_k) \frac{l}{\bar{v}^{\text{ref}} T_s} \right), \tag{2.4}$$

with $k \in [k_1, k_2]$.

Firstly, focussing on circle trajectories. With $s_{k+1} = s_k + T_s \bar{v}^{\text{ref}}$ and (2.4), $\psi_{k+1} = \int_{s_k}^{s_{k+1}} \frac{1}{R} d\eta + \psi_k = \frac{1}{R}(s_{k+1} - s_k) + \psi_k$, and consequently $\delta_k = \tan^{-1}\left(\frac{l}{R}\right)$. Transitioning from a straight with $\delta_{k_1 - 1} = 0$ to a curve at time $k_1 T_s$ (and analogously from a curve to a straight at time $k_2 T_s$), and in view of the two aforementioned constraints on δ_k, theoretical lower bounds on the turning radius are obtained as $R(k) = l / \tan(\delta^{\max})$, for $k \in \{k_1, \ldots, k_2 - 1\}$, and $R(k) = l / \tan(\dot{\delta}^{\max} T_s)$, for $k \in \{k_1 - 1, k_2\}$. For $l = 3\text{m}$, $\delta^{\max} = 35^\circ$, $T_s = 0.1\text{s}$ and $\dot{\delta}^{\max} = 25^\circ/\text{s}$, we have $R(k_1 - 1) = 68.7\text{m}$ and $R(k_1) = 4.3\text{m}$. Since the former bound is reached at only two sampling instances, it is typically neglected in practice. Nevertheless, its influence on achievable tracking errors in closed-loop is not obvious.

Analogously, and following (2.3f) and (2.4), a clothoid-segment can



(a) Circle-segment based 90°-transition with $(R, \lambda) = (6, 0.99)$.

(b) Clothoid-segment based 90°-transition with $(R, \lambda) = (8, 0)$.

Figure 9: Closed-loop tracking results. Note the correlation between tracking error and rate constraint violation of the reference (red dotted). The constraint limits are indicated by the dashed gray horizontal lines.

be defined by

$$\delta_k = \tan^{-1}\left(\left(s_k + \frac{T_s \bar{v}^{\text{ref}}}{2}\right)\sigma l\right), \tag{2.5}$$

with $k \in \mathcal{K} = \{k : s_k \text{ is element of a clothoid path-segment}\}$ and σ computed from (2.3a)-(2.3e). In contrast to the circle-case, the argument of $\tan^{-1}(\cdot)$ in (2.5) is now linearly dependent on the covered path s_k.

For (2.5), because of the missing analytical solution to (2.3d), an analytical correspondence to $R(k)$ cannot be computed as for the circle-case. Therefore, we turn to simulations and discuss the case of a 90°-turn.

Nominal Closed-Loop Simulation Experiments for a 90-degree Turn

In the following, the nominal accuracy that can be achieved by a combination of above reference trajectory designs and using LTV-MPC for closed-loop reference tracking is analyzed. As will be shown, reference trajectory design, and, in particular, references violating or not violating steering rate constraints have a significant influence on achievable tracking error. For closed-loop experiments, the control commands returned by LTV-MPC are applied to the original nonlinear model (2.1). Then, the

system is integrated forward using Matlab's `ode23tb`. Nominal conditions are assumed with noise-free and full state feedback. Different prediction horizon lengths N were tested. For smaller N the tracking accuracy decreases. For too large N the computational cost increases while not yielding smaller tracking errors. For $T_s = 0.1$s and an average traveling speed of $v = 10$km/h, $N = 20$ was found to be a good compromise. It implies a prediction horizon of $T_s v(N+1) = 5.8$m, a total number of $Nn_u = 40$ optimisation variables, and $4Nn_u = 160$ inequality constraints. The absolute tracking error is defined as $e_k^{\text{abs}} = \sqrt{(x_k - x_k^{\text{ref}})^2 + (y_k - y_k^{\text{ref}})^2}$ and is reported in unit of centimetres.

Closed-loop results are summarised Figure 9. See [152] for the detailed discussion. Several observations can be made. The smallest nominal maximal tracking error, a remarkable 0.5cm, can be achieved for the purely *clothoid*-based solution with $(R, \lambda) = (8, 0)$, where the reference path complies with steering reference *rate* constraints. Thus, the main reason for the difference in tracking errors for the two examples in Figure 9 is the compliance or violation of reference trajectories w.r.t. steering rate constraints.

In a MPC-setting, it is unavoidable to conduct interpolations such that reference trajectories begin at a given current state. Given the reference position coordinates, the reference angles can be computed as

$$\psi_k^{\text{ref}} = \tan^{-1}\left((y_{k+1}^{\text{ref}} - y_k^{\text{ref}})/(x_{k+1}^{\text{ref}} - x_k^{\text{ref}})\right).$$

Then, δ_k^{ref} can be calculated from (2.4) by invoking another $\tan^{-1}(\cdot)$. This quickly incurs jaggedness in the δ^{ref}-trajectory. It is the reason why solutions tuned to precisely meet steering rate constraints (not displayed, see [152]) in general performed worst. This is since after aforementioned interpolations steering rate limits are regularly violated by the reference trajectory δ^{ref}. Ultimately, eventhough *circle*-segment smoothing is guaranteed to *always* violate the δ^{ref}-rate constraint, it does so only for a very short period of time. Nevertheless, this short violation still has a notable effect on tracking error as Figure 9(a) illustrates.

Comment

Finally, for comparison to above case study with smoothed reference generation by means of circle- and clothoid paths fitted for a $90°$-turn, reconsider also the examples in Figure 4 and 5 when tracking an *edgy unsmoothed* $90°$-turn. In both cases, closed-loop tracking performances are unsuitable for high-precision. In Figure 4, overshoots result. In Figure 5, a steady-state tracking error can be observed. These two examples further illustrate the importance of "suitable" *reference trajectories* and the notable effect they can have on overall control performance.

SUMMARY 3. *When MPC is used for reference tracking, the overall closed-loop tracking performance is often largely dependent on the underlying reference trajectory design.*

2.1.3 Time or Spatial Parametrization

Three System Representations: Two Time- and one Spatial-based

Based on the findings from Section 2.1.1, the remainder of Section 2.1 is focused on LTV-MPC and its formulations. First, however, the question is addressed whether to use a *time-* or *spatial*-based system parametrization. This is motivated by the fact that motion planning of agile vehicles is closely related to navigation within a spatially defined environment.

Time-parameterized system dynamics in (1.2) can be transformed to a *spatial* system representation. Time and spatial derivatives shall be denoted by $\dot{x} = \frac{dx}{dt}$ and $x' = \frac{dx}{ds}$, respectively. Then, the relation

$$x' = \frac{dx}{ds} = \frac{dx}{dt}\frac{dt}{ds} = \frac{\dot{x}}{\dot{s}}$$

holds, where $\dot{s}$ denotes the velocity along some reference path. Thus, s denotes the distance along a reference path, e.g., along the road centerline.

Therefore, the nonlinear kinematic bicycle model (2.1) can be represented equivalently in a road-aligned coordinate system. In accordance with Figure 3, we obtain

$$\begin{bmatrix} \dot{s} & \dot{e}_y & \dot{e}_\psi \end{bmatrix}^\mathsf{T} = \begin{bmatrix} \frac{\rho_s v \cos(e_\psi)}{\rho_s - e_y} & v\sin(e_\psi) & \dot{\psi} - \dot{\psi}_s \end{bmatrix}^\mathsf{T}, \tag{2.6}$$

where ρ_s and $\dot{\psi}_s$ denote the road radius of curvature and road centerline heading rate, respectively. Then, in a second step, expressing $e'_\psi = \frac{\dot{e}_\psi}{\dot{s}}$ and $e'_y = \frac{\dot{e}_y}{\dot{s}}$, the *spatially* parameterized equivalent of (2.6) is derived as

$$\begin{bmatrix} e'_\psi & e'_y \end{bmatrix}^\mathsf{T} = \begin{bmatrix} \frac{(\rho_s - e_y)\tan(\delta)}{\rho_s l \cos(e_\psi)} - \psi'_s & \frac{\rho_s - e_y}{\rho_s}\tan(e_\psi) \end{bmatrix}^\mathsf{T}. \tag{2.7}$$

Note that (2.7) is entirely independent of vehicle speed v. This is characteristic for *kinematic* models, but not the case for *dynamic* vehicle models as will be further discussed below.

To summarize, there are three system representations: (2.1), (2.6), and (2.7) The latter two require a reference path, which is typically the road centerline.

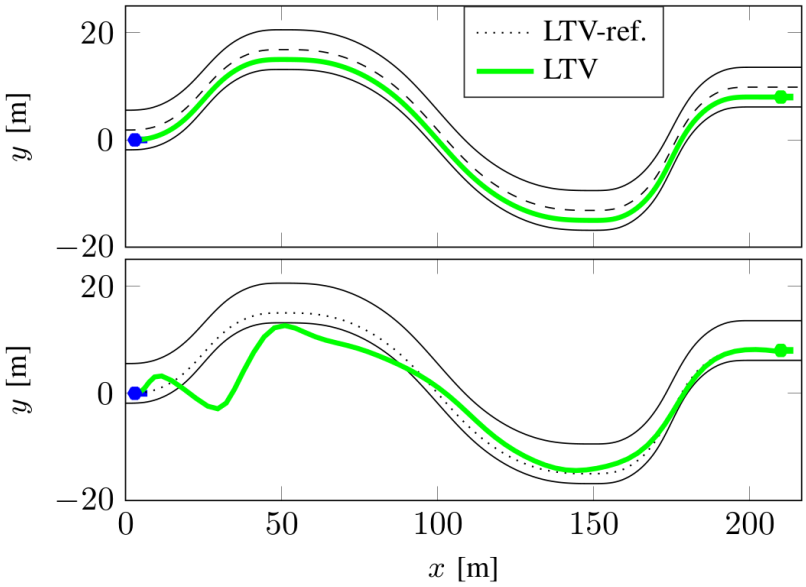


Figure 10: Example 1 of Section 2.1.3, see [148] for technical details. For indication of a two-lane road, the separating strip is given by the black dashed line. For clarity, it is omitted from the bottom plot. Top and bottom plot indicate the tracking results of the solution of a LTV-MPC optimization problem. *Reference speeds* are set as $v^{\text{ref}} = 50\text{km/h}$ (top) and $v^{\text{ref}} = 120\text{km/h}$ (bottom), respectively. The reference path (LTV-ref.) can be tracked quasi-perfectly only for the former example. Since the reference path and all other parameters are identical in both examples, the lane departure is caused solely by the difference in reference speed.

The Disadvantages of a Time-based Parametrization: Two Examples

In the following, properties of the three system representations are illustrated by means of examples. The first example is based on a *time*-parameterized LTV-MPC formulation of (2.1). For more details (the optimization problem formulation is slightly more complicated than (2.2)) the reader is referred to the extended arXiv version of [148]. The key results, which are relevant for the discussion of *time* and *spatial* parametrization, are summarized in Figure 10. Two remarks are made. First,

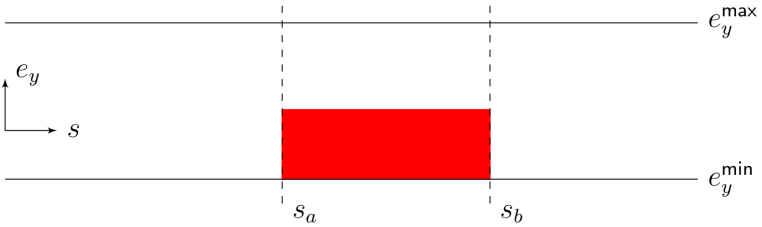


Figure 11: Example 2 of Section 2.1.3. A road corridor with an obstacle (red rectangle) is displayed. The road corridor shall be bounded by $e_y \in [e_y^{\min}, e_y^{\max}]$. The lateral obstacle width shall be denoted by $e_y \in [e_y^{\min}, e_y^{\text{obst}}]$.

the importance of *reference speed* for *time-based* MPC is underlined. Even-though a very suitable reference path is provided, namely, the obstacle-free lane centerline, safe operation is still not guaranteed. Unsuitable reference speeds can cause the vehicle to catastrophically depart from within lane boundaries. A reference speed identical to the vehicle's initial speed of 50km/h resulted in perfect reference tracking. In contrast, for $v^{\text{ref}} = 120\text{km/h}$, the vehicle departed from the road. Note that such a departure occured earliest already for $v^{\text{ref}} = 94\text{km/h}$. Second, for the given example, the catastrophic lane departure can be avoided by formulating a *time* parameterized LTV-MPC based on the road-aligned system representation(2.6) and adding constraints

$$e_y^{\min} \leq e_y(s(t)) \leq e_y^{\max}, \ \forall t, \tag{2.8}$$

where $e_y^{\min}$ and $e_y^{\max}$ denote *time-invariant* lane-boundaries.

However, to illustrate that time-parameterized MPCs are still problematic even when using (2.6) and (2.8), a second example is given. It is visualized in Figure 11. Here, the obstacle-free and spatially varying driving corridor is described by

$$e_y(s(t)) \leq e_y^{\max}, \qquad \forall s(t) \geq 0, \tag{2.9}$$

$$e_y(s(t)) \geq \begin{cases} e_y^{\min}, & \text{for } s(t) \in [0, s_a] \wedge [s(t) > s_b), \\ e_y^{\text{obst}}, & \text{for } s(t) \in [s_a, s_b]. \end{cases} \tag{2.10}$$

Suppose a linearized and discretized LTV-MPC formulation with a *reference velocity* such that for a time-index $\tilde{k} \in \{0, \ldots, K-1\}$ the reference is

$s_{\tilde{k}}^{\text{ref}} > s_b$. Then, the corresponding corridor constraint reads

$$e_y^{\min} \leq e_{y,\tilde{k}} \leq e_y^{\max}. \tag{2.11}$$

Now, suppose the *actual* state cannot track the reference accurately, such that $s_{\tilde{k}} \in [s_a, s_b]$ and $e_{y,\tilde{k}} < e_{y,\tilde{k}}^{\text{obst}}$ results. Then, this state perfectly satisfies constraint (2.11) according to the LTV-MPC formulation. However, it actually indicates a crash with the obstacle.

Several remarks are made. First and to summarize the previous example, because of the time parametrization, for the formulation of linearly constrained convex optimization problems, *time-varying* polyhedral constraints have to be defined. For a system description in absolute coordinates according to (2.1), an efficient logic for the design of such constraints that is not too conservative (too small polyhedra), and at the same time automated and sufficiently simple for applicability on arbitrarily curved road shapes is far from trivial. Second, for the road-aligned but time-based system description (2.6), the simplest corresponding formulation of polyhedral constraints reads: $s_k^{\min} \leq s_k \leq s_k^{\max}$ and $e_{y,k}^{\min} \leq e_{y,k} \leq e_{y,k}^{\max}$, $\forall k = 1, \ldots, K$, with parameters $s_k^{\min}, s_k^{\max}, e_{y,k}^{\min}$ and $e_{y,k}^{\max}$ defining the admissible road segment at each time-index k. Note, because of the time parametrization, the formulation is again strongly dependent on suitable and potentially time-varying reference velocities, that directly translate into the selection of aforementioned parameters. Furthermore, if formulated as hard state constraints the corresponding LTV-MPC optimization problem may be infeasible. Thus, polyhedral constraints not only increase the number of inequality constraints but also require more slack variables for the softening of state constraints. Finally, a remark to reference velocity generation. In [360], it is distinguished between four piecewise-affine (PWA) design methods: constant, linear, linear ramp and trapezoidal. A design logic is required to select one, *and* to additionally determine suitable slope rates and velocity plateau levels that are dependent on the current vehicle state. Thus, for *time-based* methods three tasks are required: reference path planning, reference velocity planning and reference tracking. This is difficult and results are highly heuristic and tuned.

The Benefits of a Spatial Parametrization

In contrast, within the spatial modeling framework (2.7) (where time is eliminated as the dependent variable), the lateral admissible deviation is defined at every road centerline coordinate, thereby forming a convex driving corridor which, in general, is spatially varying along the road. Importantly, in a *static* and *quasi-static*[1] obstacle environment setting, corridor constraints hold *time-invariantly*. Thus, the lateral admissible deviation is valid independently of *when* and at *what speed* a spatial coordinate and the corresponding lateral driving space is passed.

In addition, note also that the number of states in (2.7) is reduced when compared to (2.6). This naturally results from the coordinate transformation, i.e., the division of all state dynamics by $\dot{s}$.

Therefore, for spatially parameterized system models, a *linear space-varying model predictive control* (LSV-MPC) approach is taken. It is analogous to linear time-varying MPC (LTV-MPC), except for the spatial parametrization. Accordingly, linearization and discretization schemes from Section 1.3 are also adapted. To distinguish between a time and spatial parametrization, subscripts "$_k$" and "$_j$" are employed, respectively.

Later within this section, four specific aspects for the *spatial* system parametrization are discussed: a method to time schedule the reaching of waypoints along a path and the time-optimal traversal of road segments, the relation between sampling time T_s and discretization space D_s, control rate constraints in the spatial framework, and the handling of vehicle dimensions in the spatial domain for tight maneuvering.

SUMMARY 4. *For motion planning of self-driving vehicles by MPC, a spatial-based LSV-MPC formulation is preferred over a more standard time-based LTV-MPC formulation. The reason is that motion planning by time-parameterized LTV-MPC can already easily fail in static and quasi-static obstacle environments.*

[1]Here, "quasi-static" implies that a velocity offset (i.e., a simple coordinate transformation) permits to reduce the dynamic obstacle environment to a static one. It implies that dynamic obstacles move uniformly.

2.1.4 Road Modeling in the Spatial Framework

We assume that a finite number of concatenated road coordinates is available, $x_{s,i}$, $y_{s,i}$ for $i = 0, 1, \ldots, N^{\text{road}}$. Then, the corresponding distance along the road is $d_{s,i} = d_{s,i-1} + \sqrt{(x_{s,i} - x_{s,i-1})^2 + (y_{s,i} - y_{s,i-1})^2}$, initialized with some $d_{s,0}$. In order to treat the distance along the road as the dependent parameter s, we either set $s_i = d_{s,i}$ or interpolate to any arbitrary grid (e.g., uniformly spaced). The radius of curvature $\rho_s(s)$ at position s along the road centerline is computed analytically as $\rho_s(s) = \frac{\left(\left(x_s(s)'\right)^2 + \left(y_s(s)'\right)^2\right)^{3/2}}{(y_s(s)'' x_s(s)' - x_s(s)'' y_s(s))}$. By forward finite differences we can approximate $x_s(s)' \approx \frac{x_s(x+h) - x_s(s)}{h}$, $x_s(s)'' \approx \frac{x_s(s+h) - 2x_s(s) + x_s(s-h)}{h^2}$, where h is the step size. We further approximate $\psi_s(s) \approx \arctan\left(\frac{y_s(s+h) - y_s(s)}{x_s(s+h) - x_s(s)}\right)$ and $\psi_s(s)' \approx \frac{\psi_s(s+h) - \psi_s(s)}{h}$. Finally, for numerical stability we saturate $\rho_s(s) = 10^8$ (this corresponds to a drift of 5mm on 1km straight road), if $\rho_s(s) \geq 10^8$, to account for straight roads where $\rho_s(s) \to \infty$.

2.1.5 Dynamic and Kinematic Vehicle Models

It can be distinguished between two major classes of vehicle models. These are *dynamic* and *kinematic* vehicle models.

The former implies two core characteristics: a) reliance on a vehicle tire model, and b) exhibiting a singularity at zero velocity. Therefore, an additional second vehicle model for low-velocity operation is required in order to handle the entire velocity range, see [155]. The second vehicle model is typically a kinematic one.

Kinematic models do not exhibit aforementioned singularity at zero velocity. Therefore, one kinematic vehicle model can handle the entire velocity range. In order to still account for fricition and constraints on the maximal admissible velocity within tire friction limits, *two* spatial MPCs are solved in [148]. Therefore, the trajectory obtained from the first MPC solution is used to algebraically compute an upper bound on the maximal permissible velocity within tire friction limits. Therefore, a friction circle model parameterized by friction coefficient μ and the path curvature returned by the first spatial MPC solution are accounted for.

Exemplary Dynamic Vehicle Model

In [155], we consider a nonlinear vehicle model, the so-called "bicycle model", involving eight states[2], $z^t = [x,\ y,\ \psi,\ v_x,\ v_y,\ \omega,\ \omega_f,\ \omega_r]^\mathsf{T}$, and four control inputs, $u^t = [u_t,\ u_b,\ u_\delta,\ u_g]^\mathsf{T}$, whereby x and y indicate the center of gravity (CoG) in the inertial frame, ψ denotes the yaw angle (relative to the inertial frame), the longitudinal and lateral velocities are given by v_x and v_y, and where ω, ω_f and ω_r are yaw rate, front and rear wheel speed, respectively. The inputs u_t and u_b denote throttle and brake pedal positions. The front steering input is u_δ and the integer transmission gear control variable is denoted by $u_g \in \{1,\ 2,\ 3,\ 4\}$. Thus, dropping the arguments and dependencies for brevity, the time-dependent exem-

[2]The superscript "t" is here used to indicate time parametrization.

plary *dynamic* vehicle model used in [155] is:

$$\dot{x} = v_x \cos(\psi) - v_y \sin(\psi), \tag{2.12a}$$

$$\dot{y} = v_x \sin(\psi) + v_y \cos(\psi), \tag{2.12b}$$

$$\dot{\psi} = \omega, \tag{2.12c}$$

$$\dot{v}_x = \frac{1}{m}\big(F_{f,x}\cos(u_\delta) + F_{r,x} - F_{f,y}\sin(u_\delta) - F_r\big) + \omega v_y, \tag{2.12d}$$

$$\dot{v}_y = \frac{1}{m}\left(F_{f,x}\sin(u_\delta) + F_{f,y}\cos(u_\delta) + F_{r,y}\right) - \omega v_x, \tag{2.12e}$$

$$\dot{\omega} = \frac{1}{I_z}\left(a(F_{f,x}\sin(u_\delta) + F_{f,y}\cos(u_\delta)) - bF_{r,y}\right), \tag{2.12f}$$

$$\dot{\omega}_f = \frac{1}{I_\omega}\left(T_d - T_b - r_t F_{f,x}\right), \tag{2.12g}$$

$$\dot{\omega}_r = \frac{1}{I_\omega}\left(T_d - T_b - r_t F_{r,x}\right), \tag{2.12h}$$

where $F_{f,x}(v_x, u_\delta, v_y, \omega, \omega_f)$ denotes the longitudinal force on the front tire. It is a non-smooth nonlinear function, modeled here as a look-up table calibrated experimentally. $F_{r,x}(v_x, \omega_r)$ is the force acting longitudinally on the rear wheel, which is also modeled by a look-up table. The lateral forces on the front and rear wheels are denoted by $F_{f,y}(u_\delta, v_y, \omega, v_x)$ and $F_{r,y}(v_x, v_y, \omega)$. The engine is modeled by a two-dimensional look-up table, that maps engine speed and throttle input to engine torque. The drive torque $T_d(u_t, \omega_f, u_g)$ is ultimately computed from engine torque via an algebraic relation. The brake torque $T_b(u_b)$ is modeled as a linear function of the brake input signal. The vehicle mass, yaw and wheel inertia are indicated by m, I_z and I_ω, respectively. The quantities r_t, a and b denote the tire radius, distance of CoG from front and rear axle, respectively. We compactly summarize the system model (2.12) as $\dot{z}^t = f^t(z^t, u^t)$. It is further visualized in Figure 12.

In the LSV-MPC formulation of [155], to *encourage* stability we added soft constraints on front and rear tire slip angles to not enter the (strongly) nonlinear/unstable region of tire characteristics [106], [211]. For the provided vehicle model, lateral forces on front and rear tires, $F_{f,y}(u_\delta, v_y, \omega, v_x)$ and $F_{r,y}(v_x, v_y, \omega)$, are given as a linear function of slip angles $\alpha_f(\omega, v_y, v_x)$ and $\alpha_r(\omega, v_y, v_x)$ with subsequent saturations at maximal lateral forces.

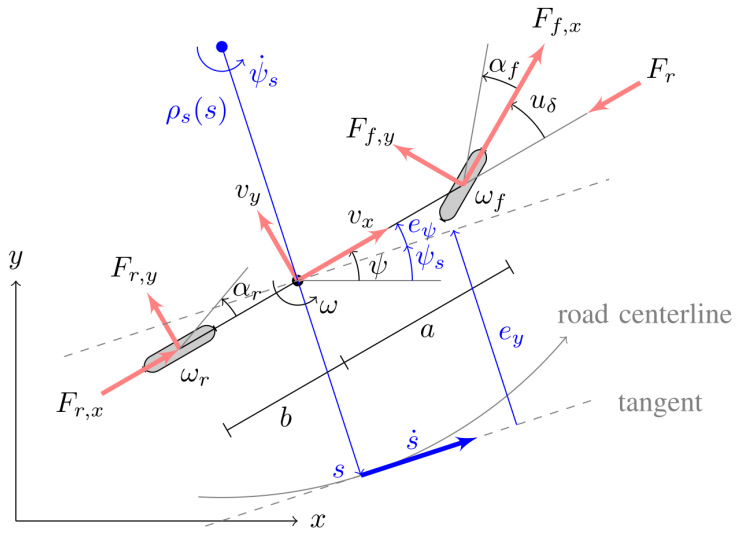


Figure 12: The nonlinear dynamic bicycle model including the representation of the curvilinear coordinate system used in [155].

Note that in contrast to the aforementioned two references [106], [211], linearization and discretization need to be carried out in *space* coordinates when deriving LSV-MPC formulations as in our case.

Extension to the Whole Operating Range for Dynamic Vehicle Models

For the *dynamic* system description, a singularity for $v = \sqrt{v_x^2 + v_y^2} = 0$ is characteristic. Let us now consider a *kinematic* vehicle model lacking de/acceleration dynamics. The classic nonlinear kinematic bicycle model [317] is stated in (2.1). As discussed in Section 2.1.3, the spatial equivalent (2.7) is entirely independent of vehicle speed v. Thus, speed and steering are naturally decoupled as a result of the spatial transformation. This is relevant since the whole operating range (speed) of automated vehicles can now be served. Kinematic time-dependent vehicle models exhibit a linear dependency on velocity v. This dependency is always eliminated through the transformation to a spatial-based coordinate system. Consider the transformation $\frac{d(\cdot)}{ds} = \frac{d(\cdot)}{dt}\frac{1}{\dot{s}}$ and further that $\dot{s}$

is proportional to v for kinematic vehicle models. Thus, employing one dynamic and one kinematic vehicle model for higher speeds and velocities close to 0, respectively, permits control design entirely spatial-based with state-transitions being space- instead of time-varying.

The Role of Velocity and Friction for Kinematic Vehicle Models

The spatial transformation eliminates any velocity dependence of a kinematic vehicle model expressed in the road-aligned coordinate system. In order for the trajectory planning method to still provide velocity information, we employ the method from [120, Sect. 3] to determine *spatially varying* upper bounds, $v^{\text{max,fric}}(s)$, on admissible traveling speeds. Then, in [156], [148] these speed limits are imposed as constraints on velocity in a *sequential* iteration as discussed next. Besides the gravitational acceleration constant, a friction coefficient μ must be assumed. Then, any reference traveling speeds $v^{\text{ref}}(s) \leq v^{\text{max,fric}}(s)$ are consequently within vehicle tire friction limits.

SUMMARY 5. *The deployment of a dynamic or a kinematic vehicle model for motion planning should be dependent on the application. However, for low velocity operation a kinematic vehicle model must be employed since dynamic vehicle models exhibit a singularity at zero velocity.*

SUMMARY 6. *Parameters such as the friction coefficient μ typically vary during online operation for different road surfaces and weather conditions. Therefore, there are two fundamental approaches. First, varying parameters can be treated as external vehicle model disturbances and incorporated directly within the MPC formulation. Secondly and alterantively, leveraging a mathematical description of underlying physics, varying parameters can also be used to compute, outside of MPC, a) state constraints or b) bounds on reference setpoints, which can consequently be used for the MPC formulation. An example is to compute a limit on maximal admissible velocity within tire friction limits as a function of μ and the road curvature of the road segment ahead.*

For MPC, Summary 6 is especially relevant for *kinematic* vehicle models. However, by suitable adaptation and filtering of *setpoints* on the permissible traveling velocity, it also is especially relevant for the neural network control approach discussed in Section 2.2, which is based on the idea of encoding motion primitives.

2.1.6 Sequential Programming

In the next Section 2.1.7, it is discussed how the LSV-MPC formulation can incorporate velocity as a control variable also for kinematic vehicle models. Therefore, two options were considered to also incorporate friction constraints. The first was motivated by [273], where a term penalizing lateral accelerations was added to the cost function in order for the vehicle to automatically slow down in anticipation of tight turns. We approximated lateral acceleration as $a_y = v\dot{\psi} = \frac{v^2 \tan(\delta)}{l}$ [369], conducted a linearization, and incorporated a minmax-type penalty in the cost function. This method has three (significant) disadvantages: a) references v_j^{ref} and δ_j^{ref}, $\forall j = 0, \dots, N$ are required for computation, b) it is not obvious how to *weight* said penalities, and c) the formulation does *not* guarantee operation within friction limits. We therefore instead opt for a second option, and thus conduct the following two-step algorithm for kinematic vehicle models:

1. Solve a LSV-MPC problem to obtain *at least* $\{e_{\psi,j}\}_{j=0}^{N}$, $\{e_{y,j}\}_{j=0}^{N}$, $\{v_j\}_{j=0}^{N-1}$, $\{\delta_j\}_{j=0}^{N-1}$ expressed along the spatial discretization grid $\{s_j\}_{j=0}^{N}$, where s_j is abbreviated for $s_{\tau+j}$ when planning at time τ.

2. Use the trajectories of Step 1 as references for a second solution of a LSV-MPC problem with *additional* constraints

$$v_j \leq v_j^{\text{max,fric}}, \ \forall j = 0, \dots, N-1, \tag{2.13}$$

   where $v_j^{\text{max,fric}}$ is computed according to [121, Sect. 3], assuming a friction coefficient μ, and denoting the maximum admissible velocity permitting operation within vehicle tire friction limits.

The first step is to generate a suitable vehicle trajectory based on which $v_j^{\text{max,fric}}$ can be computed. The second step is to refine velocity control. Therefore, by definition a sequential programming approach is proposed. In [156] and [148], sequential *linear* programming (SLP) is proposed.

Note that the incorporation of friction constraints according to (2.13) is foremostly relevant for *kinematic* vehicle models. In contrast, the approach for the LSV-MPC formulation for *dynamic* vehicle models is different. In [155], stability is encouraged by adding soft constraints on front

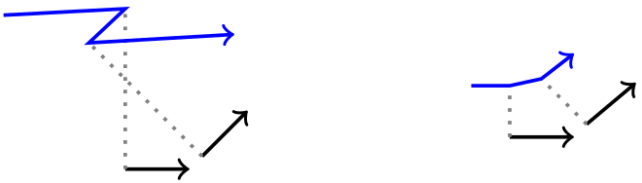


Figure 13: Sketching the motivation for SP-iterations. Large deviations between the trajectory output from the LSV-MPC solution and the reference trajectory can result in jaggedness (left). This undesired jaggedness can be alleviated by solving LSV-MPC problems repeatedly causing optimization and reference trajectory to converge iteratively (right).

and rear tire slip angles to not enter the (strongly) nonlinear/unstable region of tire characteristics.

Nevertheless, the SLP-approach can offer advantages for both kinematic and dynamic vehicle models. For obstacle constellations requiring larger steering maneuvers, transition dynamics and vehicle dimension constraints are strongly dependent on underlying reference trajectories used for linearization and discretization. This is of particular relevance for the first initialization of references, for which we reconstruct $\{e_{\psi,j}^{\text{ref}}\}_{j=0}^{N}$ and $\{e_{y,j}^{\text{ref}}\}_{j=0}^{N}$ from a least-heading-varying PWA path avoiding all obstacles. Therefore, we propose the sequential programming (SP) approach. See also Figure 13 for visualization.

SUMMARY 7. *For general motion planning of automated vehicles by LSV-MPC, a sequential programming approach is proposed. For kinematic vehicle models, this is motivated to introduce velocity bounds to keep within vehicle tire friction limits. In general, for both dynamic and kinematic vehicle models, this is motivated to iteratively let optimization output and reference, i.e., the optimization output from the previous sequential iteration, converge.*

SUMMARY 8. *For simple road centerline tracking applications, a sequential programming approach is not necessarily required. The road centerline curvature of the road segment ahead can be used equivalently to compute velocity bounds to keep operation within vehicle tire friction limits.*

2.1.7 Time Scheduling in the Spatial Framework

The *spatial* parametrization permits to express time t as a function of space coordinate s. Based on a kinematic bicycle model (the dynamic case is discussed further below), we therefore relate $t' = \frac{1}{\dot{s}}$ and obtain

$$t' = \frac{\rho_s - e_y}{\rho_s v \cos(e_\psi)}. \tag{2.14}$$

Throughout the following, it is assumed $e_\psi \in (-\frac{\pi}{2}, \frac{\pi}{2})$, $\rho_s > e_y$ and $v > 0$. This is made to avoid poles at $e_\psi = \pm\frac{\pi}{2}$ in (2.7) and (2.14). Let the linearization of (2.14) along the discretization grid be denoted by $t_{j+1} = t_j + a_{t,j} z_j + b_{t,j} u_j + g_{t,j}$.

Property 1. *For timely mission planning (scheduling) and the formulation of convex optimization problems with linear constraints, we require* $t_{j+l} > t_j,\ \forall l > 0$.

Proposition 1. *The linearized dynamics of* (2.14), $t_{j+1} = t_j + a_{t,j} z_j + b_{t,j} u_j + g_{t,j}$, *violate Property 1.*

Proof. We express $t_{j+1} = t_j + a_{t,j}^{e_\psi}(e_{\psi,j} - e_{\psi,j}^{\text{ref}}) + a_{t,j}^{e_y}(e_{y,j} - e_{y,j}^{\text{ref}}) + b_{t,j}^{v}(v_j - v_j^{\text{ref}}) + f_j^{\text{ref}}$ with $f_j^{\text{ref}} = f(z_j^{\text{ref}}, u_j^{\text{ref}})$. We have $b_{t,j}^{v} = -\frac{\rho_j^{\text{ref}} - e_{y,j}^{\text{ref}}}{\rho_j^{\text{ref}} (v_j^{\text{ref}})^2 \cos(e_{\psi,j}^{\text{ref}})}$, which *always* is negative. Then, $f_j^{\text{ref}} + b_{t,j}^{v}(v_j - v_j^{\text{ref}}) = \frac{\rho_{s,j}^{\text{ref}} - e_{y,j}^{\text{ref}}}{\rho_{s,j}^{\text{ref}} v_j^{\text{ref}} \cos(e_{\psi,j}^{\text{ref}})}(2 - \frac{v_j}{v_j^{\text{ref}}})$. To prove the proposition, it suffices to find one counterexample violating $t_{j+1} > t_j$. Such can be constructed by $e_{\psi,j} = e_{\psi,j}^{\text{ref}}$, $e_{y,j} = e_{y,j}^{\text{ref}}$ and $v_j > 2v_j^{\text{ref}}$, which concludes the proof. □

Remark 3. *Proposition 1 is also valid when simplifying* (2.14) *to* $t' = 1/v\cos(e_\psi)$ *under the assumption of* $\rho_s \gg e_y$. *It is further also valid for a small* e_ψ*-angle approximation with* $\cos(e_\psi) \approx 1 - \frac{e_\psi^2}{2}$. *This is since counterexamples can be constructed similarly to before.*

Proposition 1 and Remark 3 have important implications. A minimization problem with objective $\min\{t_N\}$ can result in optimal solution $\min\{t_N\} < 0$, thereby decoupling variable t from physical interpretation and thus making it useless for time scheduling tasks.

Proposition 2. *The coordinate transformation* $q_j = \frac{1}{v_j}$ *and dynamics*

$$t_{j+1} = t_j + \frac{(s_{j+1} - s_j)(\rho^{ref}_{s,j} - e^{ref}_{y,j})}{\rho^{ref}_{s,j} \cos(e^{ref}_{\psi,j})} q_j, \tag{2.15}$$

approximate the linearization of (2.14) *for finite reference velocity and satisfy Property 1.*

Proof. Under the coordinate transformation and abbreviating $D_{s,j} = s_{j+1} - s_j$, the discrete equation of (2.14) is $t_{j+1} = t_j + D_{s,j} \frac{\rho_{s,j} - e_{y,j}}{\rho_{s,j} \cos(e_{\psi,j})} q_j$, and linearized $t_{j+1} = t_j + a^{e_\psi, q}_{t,j}(e_{\psi,j} - e^{\text{ref}}_{\psi,j}) + a^{e_y, q}_{t,j}(e_{y,j} - e^{\text{ref}}_{y,j}) + b^q_{t,j} q_j$. Note that $a^{e_\psi, q}_{t,j}$ and $a^{e_y, q}_{t,j}$ are proportional to q^{ref}_j and can be positive- *or* negative. Thus, to guarantee $t_{j+1} > t_j$, they must be eliminated from t_{j+1}-dynamics. This is achieved for $q^{\text{ref}}_j \to 0$ (i.e., $v^{\text{ref}}_j \to \infty$) by proportionality. The remainder yields (2.15), which is the exact linearization of (2.14) for $v^{\text{ref}}_j \to \infty$, and approximate for finite reference velocities. □

Importantly, the coordinate transformation does not affect the linearized and discretized state dynamics for z_{j+1}. This is since they are independent of v_j, and consequently also independent of q_j.

So far, the discussion focused on *kinematic* vehicle models. Let us extend the discussion to *dynamic* vehicle models [225], [155]. The equivalent to (2.14) can be derived analogously, resulting in

$$t' = \frac{\rho_s - e_y}{\rho_s \left(v_x \cos(e_\psi) - v_y \sin(e_\psi)\right)}, \tag{2.16}$$

where v_x and v_y denote longitudinal and lateral velocities relative to the inertial vehicle frame. Importantly, v_x and v_y are vehicle states and not control variables anymore.

Proposition 3. *The linearized dynamics of* (2.16)*, of the form* $t_{j+1} = t_j + a_{t,j} z_j + b_{t,j} u_j + g_{t,j}$*, violate Property 1.*

Proof. Similar to the one of Proposition 1. □

Additional complexity arises from state variable v_y, that is absent in the kinematic case. In practice, typically $v_y \ll v_x$.

Proposition 4. *The coordinate transformation* $q_j^{v_x} = \frac{1}{v_{x,j}}$ *and dynamics*

$$t_{j+1} = t_j + \frac{(s_{j+1} - s_j)(\rho_{s,j}^{ref} - e_{y,j}^{ref})}{\rho_{s,j}^{ref} \cos(e_{\psi,j}^{ref})} q_j^{v_x}, \tag{2.17}$$

approximate the linearization of (2.14) *and satisfy Property 1.*

Proof. Similar to the one of Proposition 2. □

The time dynamics (2.17) are characteristic for the dynamic vehicle model case in the sense that state v_y is omitted entirely from consideration. This is done to comply with Property 1. Note that according to the coordinate transformation, $q_j^{v_x} = \frac{1}{v_{x,j}}$, also all state equations (and consequently linearization and discretization routines) need to be updated. Based on (2.17), QPs and LPs for LSV-MPC can now be formulated.

Comment

Note that both dynamic and kinematic bicycle models are two representations of the same real-world vehicle. In this perspective, above mentioned variations and approximations characterize other different model representations of the real-world vehicle. Note that any model variations are legitimized as long as physical actuator absolute and rate constraints are not modified and as long as resulting model variation can serve a desired application.

SUMMARY 9. *The spatial framework permits to express time as a function of the dependent spatial coordinate* s*. This enables to formulate time scheduling constraints (e.g., useful at intersections) and an objective functions for minimum time traversal of road segments (e.g., useful for racing). It also enables velocity control when using kinematic vehicle models for LSV-MPC. Recall that for kinematic vehicle models, velocity is otherwise eliminated by the spatial coordinate transformation. Time-scheduling, and thus the spatial framework, are also considered to be particularly useful for the coordination of multi-automated vehicle systems.*

2.1.8 Sampling Times in the Spatial Framework

From a simple Euler forward discretization of (2.16) we can derive a relation between sampling time T_s and discretization space D_s as follows

$$D_s = T_s \frac{\rho_{s,0}(v_{x,0}\cos(e_{\psi,0}) - v_{y,0}\sin(e_{\psi,0}))}{\rho_{s,0} - e_{y,0}}, \tag{2.18}$$

where $\rho_{s,0}$, $v_{x,0}$, $e_{\psi,0}$, $v_{y,0}$ and $e_{y,0}$ here denote corresponding estimated or measured properties at the time of trajectory planning, i.e., when formulating the current LSV-MPC problem. Then, D_s according to (2.18) can be used to generate a uniformly spaced prediction horizon based on which the LSV-MPC problem is formulated. Thus, (2.18) enables to synchronize a fixed sampling time T_s with a spatial LSV-MPC problem formulation. Note that for centerline tracking of straight roads, it holds that $\rho_{s,0} \to \infty$, $e_{\psi,0} \to 0$ and $e_{y,0} \to 0$. Thus, we recover $D_s = T_s v_{x,0}$.

2.1.9 Control Rate Constraints in the Spatial Framework

Continuous rate constraints for control variables v and δ are of the form

$$\dot{v}^{\min} \leq \dot{v} \leq \dot{v}^{\max} \quad \text{and} \quad \dot{\delta}^{\min} \leq \dot{\delta} \leq \dot{\delta}^{\max}, \tag{2.19}$$

whereby the bounds ($\dot{v}^{\min}$, $\dot{v}^{\max}$, $\dot{\delta}^{\min}$ and $\dot{\delta}^{\max}$), in general, are nonlinear functions of the vehicle's operating point. They are time-varying parameters, for example, dependent on engine speed and torque. By applying the spatial coordinate transformation, a discretization, the change of variables according to the previous section, and assuming the bounds to remain constant for the duration of the planning horizon, we obtain

$$\frac{D_{s,j}\dot{v}^{\min}(\rho_{s,j} - e_{y,j})q_j}{\rho_{s,j}\cos(e_{\psi,j})} \leq \frac{1}{q_{j+1}} - \frac{1}{q_j} \leq \frac{D_{s,j}\dot{v}^{\max}(\rho_{s,j} - e_{y,j})q_j}{\rho_{s,j}\cos(e_{\psi,j})}, \tag{2.20}$$

$$\frac{D_{s,j}\dot{\delta}^{\min}(\rho_{s,j} - e_{y,j})q_j}{\rho_{s,j}\cos(e_{\psi,j})} \leq \delta_{j+1} - \delta_j \leq \frac{D_{s,j}\dot{\delta}^{\max}(\rho_{s,j} - e_{y,j})q_j}{\rho_{s,j}\cos(e_{\psi,j})}, \tag{2.21}$$

whereby we abbreviated $D_{s,j} = s_{j+1} - s_j$ for the general case. Thus, linear and control channel-separated rate constraints in (2.19) are rendered not only nonlinear, but additionaly state-dependent, and also velocity-dependent for steering control. This has two implications. First, to formulate linearly constrained optimization problems, we require the linearization of (2.20) and (2.21). Dependent on the quality of underlying reference trajectories, this may incur significant distortions. Second, while the discrete form of (2.19) can always be guaranteed to be feasible (assuming a feasible initialization), for (2.20) and (2.21) this is not the case anymore. Thus, slack variables are required. Let us consider two degrees of simplification of (2.20) and (2.21). First, we assume large $\rho_{s,j}$ and small $e_{\psi,j}$, and consequently approximate $\frac{(\rho_{s,j} - e_{y,j})}{\rho_{s,j}\cos(e_{\psi,j})} \approx 1$, thereby rendering (2.20) and (2.21) state-independent, but maintaining velocity-dependent bounds. Thus, steering rate constraints still depend on q_j. Second, we additionally eliminate this velocity-dependency and formulate

$$\tilde{T}_j\dot{v}^{\min} \leq \frac{1}{q_{j+1}} - \frac{1}{q_j} \leq \tilde{T}_j\dot{v}^{\max},\ j = 0, \ldots, N-2, \tag{2.22}$$

$$\tilde{T}_j\dot{\delta}^{\min} \leq \delta_{j+1} - \delta_j \leq \tilde{T}_j\dot{\delta}^{\max},\ j = 0, \ldots, N-2, \tag{2.23}$$

with

$$\tilde{T}_j = \frac{D_{s,j}}{v_j^{\max}}, \tag{2.24}$$

where $v_j^{\max}$ denotes the maximal permissible velocity within spatial segment $s \in [s_j, s_{j+1}]$. The velocity limit may be given by road speed limits and friction constraints. Then, (2.22) and (2.23) guarantee that time-based rate constraints are also respected in the spatial domain. This is since in the curvilinear coordinate system, the distance $D_{s,j} = s_{j+1} - s_j$ cannot be covered faster than with velocity $v_j^{\max}$.

Formulations (2.22) and (2.23) bear the advantage of separating control channels and are therefore our preferred form for *spatial-based* rate constraints. We denote the linearization of (2.22) by

$$c_{q,j}^{\min} \leq b_{q,j+1} q_{j+1} + b_{q,j} q_j \leq c_{q,j}^{\max},\ j = 0, \ldots, N-2. \tag{2.25}$$

SUMMARY 10. *The transformation of time-dependent control rate constraints* (2.19) *to the road-aligned coordinate frame is not trivial and to be considered as the main disadvantage of a spatial-based system representation. We opted for the "simple" control channel-seperating forms* (2.23) *and* (2.25) *in linearly constrained optimization problems, and discussed the role of $\tilde{T}_j$ as a transformation parameter. A conservative $\tilde{T}_j$ is proposed in* (2.24) *that guarantees that time-based rate constraints are guaranteed to also be respected in the spatial domain.*

2.1.10 Vehicle Dimension Constraints in the Spatial Framework

Two methods for the handling of *vehicle dimensions* are considered. These are a) the *inflation of obstacles or road bounds* [155], [148], and b) a *linearization approach* [156]. The latter is suitable for tight maneuvering but computationally more expensive.

1. Iterative Linearization Approach

We model vehicles as rectangles. This is a simple and yet an accurate vehicle representation. As illustrated in Figure 3, parameters a, b, and w indicate distances between the center of gravity (CoG) and rear, front and lateral vehicle sides, respectively. The four vehicle corners $c_i,\ i = 1,\dots,4$, can be expressed as

$$s_{c_i} = s + \xi^s_{c_i}\cos(e_\psi) + \zeta^s_{c_i}\sin(e_\psi), \tag{2.26}$$

$$e_{y,c_i} = e_y + \xi^{e_y}_{c_i}\cos(e_\psi) + \zeta^{e_y}_{c_i}\sin(e_\psi), \tag{2.27}$$

with $\xi^s_{c_i}, \xi^{e_y}_{c_i} \in \{b, -a, -a, b\}$, $\zeta^s_{c_i} \in \{-w, -w, w, w\}$, and further $\zeta^{e_y}_{c_i} \in \{w, w, -w, -w\}$ for $c_i,\ i = 1,\dots,4$, respectively.

Let us derive convex vehicle dimension constraints. At every s_j, assuming forward motion, we can describe lateral vehicle boundaries affine in s and nonlinear in $e_{\psi,j}$ as

$$e^{\text{lower}}_{y,j}(s) = \tan(e_{\psi,j})(s - s_{j,c_3}) + e_{y,c_3}, \tag{2.28}$$

$$e^{\text{upper}}_{y,j}(s) = \tan(e_{\psi,j})(s - s_{j,c_2}) + e_{y,c_2}, \tag{2.29}$$

accounting for (2.26). We define the set

$$\begin{aligned}\tilde{\mathcal{S}}_j &= \{\{s_k\}_{k=1}^{N-1} : s_j - \Delta s_{j,\min} \le s_k \le s_j + \Delta s_{j,\max}\} \\ &=: \{s_{\tilde{k}_1}, s_{\tilde{k}_2}, \dots, s_{\tilde{k}_{\tilde{N}_j}}\},\end{aligned}$$

with

$$\Delta s_{j,\min} = \min(s_{j,c_2}, s_{j,c_3}), \Delta s_{j,\max} = \max(s_{j,c_1}, s_{j,c_4}),$$

and

$$\mathcal{S}_j = \{s_{\tilde{k}_1-1}, s_{\tilde{k}_1}, \dots, s_{\tilde{k}_{\tilde{N}_j}}, s_{\tilde{k}_{\tilde{N}_j}+1}\} =: \{s_{k_1}, \dots, s_{k_{\bar{N}_j}}\}, \tag{2.30}$$

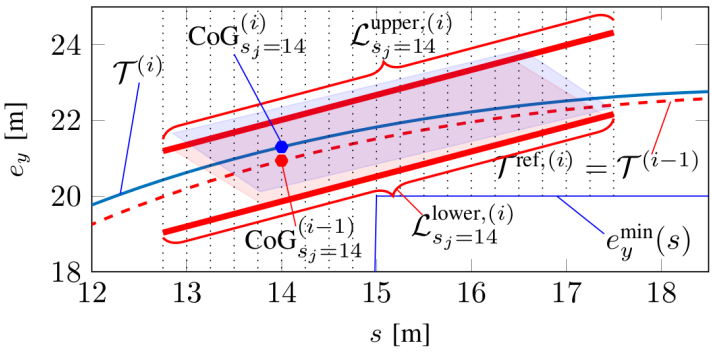


Figure 14: Illustration of vehicle dimension constraints. Indices (i) and $(i-1)$ indicate the corresponding SLP-iteration. The planning result at path coordinate $s_j = 14$m is displayed.

to also guarantee coverage of vehicle corners in between any two grid points. The linearization of (2.28) and (2.29) yields

$$e^{\mathrm{lower}}_{y,\mathrm{lin},j}(s) = \begin{bmatrix} g^{\mathrm{lower}}(s) & 1 \end{bmatrix} \begin{bmatrix} e_{\psi,j} & e_{y,j} \end{bmatrix}^{\mathsf{T}} + h^{\mathrm{lower}}_{\mathrm{lin},j}(s), \tag{2.31}$$

$$e^{\mathrm{upper}}_{y,\mathrm{lin},j}(s) = \begin{bmatrix} g^{\mathrm{upper}}(s) & 1 \end{bmatrix} \begin{bmatrix} e_{\psi,j} & e_{y,j} \end{bmatrix}^{\mathsf{T}} + h^{\mathrm{upper}}_{\mathrm{lin},j}(s), \tag{2.32}$$

with $g^{\mathrm{lower}}(s)$, $h^{\mathrm{lower}}_{\mathrm{lin},j}(s)$, $g^{\mathrm{upper}}(s)$, and $h^{\mathrm{upper}}_{\mathrm{lin},j}(s)$ parameterized by s_j, $e^{\mathrm{ref}}_{\psi,j}$, and $e^{\mathrm{ref}}_{y,j}$. The main motivation of vehicle dimension constraints is to ensure that the vehicle geometry is constrained to the interior of the road corridor. By evaluating (2.31) and (2.32) at the discrete grid points of (2.30), this can be expressed as the set of inequalities

$$\begin{bmatrix} e^{\mathrm{lower}}_{y,\mathrm{lin},j}(s_{k_1}) \\ \vdots \\ e^{\mathrm{lower}}_{y,\mathrm{lin},j}(s_{k_{\bar{N}_j}}) \end{bmatrix} \geq \begin{bmatrix} e^{\min}_{y}(s_{k_1}) \\ \vdots \\ e^{\min}_{y}(s_{k_{\bar{N}_j}}) \end{bmatrix}, \tag{2.33}$$

$$\begin{bmatrix} e^{\mathrm{upper}}_{y,\mathrm{lin},j}(s_{k_1}) \\ \vdots \\ e^{\mathrm{upper}}_{y,\mathrm{lin},j}(s_{k_{\bar{N}_j}}) \end{bmatrix} \leq \begin{bmatrix} e^{\max}_{y}(s_{k_1}) \\ \vdots \\ e^{\max}_{y}(s_{k_{\bar{N}_j}}) \end{bmatrix}. \tag{2.34}$$

We summarize the left-hand sides of the inequality signs by $\mathcal{L}^{\mathrm{lower}}_{s_j}$ and $\mathcal{L}^{\mathrm{upper}}_{s_j}$, respectively. For visualization, see Figure 14. Inequalities (2.33)

and (2.34) are linear in state z_j at position s_j and can be compactly summarized as $Q_j^{\text{lower}} z_j \geq q_j^{\text{lower}}$, and $Q_j^{\text{upper}} z_j \leq q_j^{\text{upper}}$, with $Q_j^{\text{lower}},\ Q_j^{\text{upper}} \in \mathbb{R}^{\bar{N}_j \times 2}$, $q_j^{\text{lower}},\ q_j^{\text{upper}} \in \mathbb{R}^{\bar{N}_j}$, and $\bar{N}_j$ variable for each s_j and dependent on references $e_{\psi,j}^{\text{ref}}$ and $e_{y,j}^{\text{ref}}$. Finally, note that instead of $\tilde{\mathcal{S}}_j$, as a *least-conservative* variant, it could be differentiated between two grid segments $\tilde{S}_j^{\text{lower}}$ and $\tilde{S}_j^{\text{upper}}$ that are different for both lateral vehicle sides, instead of having one $\tilde{S}_j$ common to both.

2. Obstacle and Road Boundary Inflation Approach

For the navigation of automated vehicles, especially in very constrained environments, vehicle dimensions must be accounted for. This is particularly relevant for large-sized vehicles such as heavy-duty trucks or buses. In contrast to above method based on iterative linearization of nonlinear vehicle dimension constraints, here a different approach is taken. A velocity-dependent and computationally much less demanding *heuristic* is proposed. We denote road corridor constraints as

$$e_{y,j}^{\min} + \Delta e_{y,\tau} \leq e_{y,j} \leq e_{y,j}^{\max} - \Delta e_{y,\tau},\ j = 1, \ldots, N, \tag{2.35}$$

with corridor boundaries $e_{y,j}^{\min}$ and $e_{y,j}^{\max}$, and margin $\Delta e_{y,\tau} \geq 0$ determined at time τ. For point mass trajectory planning without vehicle dimension constraints we have $\Delta e_{y,\tau} = 0$.

Proposition 5. *To guarantee safe vehicle operation within road boundaries according to* (2.35)*, assuming rectangular vehicle dimensions with $w < b$, and assuming forward motion, we require $\Delta e_{y,\tau} = \max_{e_\psi \in \mathcal{E}} \Delta e_{y,\tau}(e_\psi)$ with*

$$\Delta e_{y,\tau}(e_\psi) = b \sin(e_\psi) + w \cos(e_\psi), \tag{2.36}$$

and $\mathcal{E} = (-\pi/2, \pi/2)$.

Proof. From the definitions of b and w in Figure 3. □

Remark 4. *The selection of $\Delta e_{y,\tau}$ according to Proposition 5 is conservative. Let us denote the associated angle by e_ψ^{max}, whereby $e_\psi^{max} = \tan^{-1}(b/w)$ is derived from the maximization. For $w = 0.9$ and $b = 3.5$ used in simulations, we obtain $e_\psi^{max} = 75.6°$. Such a large deviation from the road heading is not admissible at high speeds. By tighter constraining $\mathcal{E}$, the level of conservativeness can be reduced. Note that we have $w \leq \Delta e_{y,\tau}(e_\psi) \leq \Delta e_{y,\tau}(e_\psi^{max})$ for*

$e_\psi \in [0, e_\psi^{max}]$, *and that* $\Delta e_{y,\tau}(e_\psi)$ *is strictly monotonously increasing for that heading range.*

In the following, we derive a heuristic for the selection of $\Delta e_{y,\tau}$. The ideas are a) to relate to vehicle speed, and b) to obtain monotonously increasing $\Delta e_{y,\tau}$ with increasing v. Let $v^{\max}$ denote the maximum highway speed limit (e.g., $v^{\max} = 120$km/h), and v_τ the traveling velocity at time τ. Let us first state the algorithm before discussing two variants:

1. Determine an angle $e_\psi^v \in [0, e_\psi^{\max}]$.
2. Compute $e_\psi = \frac{v_\tau}{v^{\max}} e_\psi^v$.
3. Compute $\Delta e_{y,\tau} = b\sin(e_\psi) + w\cos(e_\psi)$ and adapt corridors in (2.35).

Note that by design, both of the mentioned motivating ideas are addressed. Also, the level of conservativeness can be controlled by e_ψ^v. We considered two options. First, $e_\psi^v = e_\psi^{\max}$. Second, we note that specific vehicle velocities only admit a very limited deviation from the road heading direction to take corrective steering action within a limited "reaction time" Γ. Assuming a road boundary $\tilde{e}_y^{\max} > 0$ and a vehicle position $\tilde{e}_y < \tilde{e}_y^{\max}$, the front left vehicle corner reaching the road boundary $\tilde{e}_y^{\max}$ within time Γ can be expressed as

$$\tilde{e}_y + \Gamma v_\tau \sin(\tilde{e}_\psi) + b\sin(\tilde{e}_\psi) + w\cos(\tilde{e}_\psi) = \tilde{e}_y^{\max}, \tag{2.37}$$

where $\tilde{e}_\psi \geq 0$ denotes the vehicle heading. Thus, for our second variant, we solve (2.37) analytically for $\tilde{e}_\psi$, and set $e_\psi^v = \tilde{e}_\psi$ in Step 1). This method is less conservative/generates smaller $\Delta e_{y,\tau}$ than the first variant with $e_\psi^v = e_\psi^{\max}$. This is since $\tilde{e}_\psi < e_\psi^{\max}$ for typical parameter choices, and since $\Delta e_{y,\tau}(e_\psi)$ is strictly monotonously increasing for $e_\psi \in [0, e_\psi^{\max}]$. Note that the front left vehicle corner was considered for derivation. This is appropriate for our parameter choices of $\tilde{e}_y = |e_y(s_\tau)|$ and $\tilde{e}_y^{\max} = \min_j\{\min(e_{y,j}^{\max}, |e_{y,j}^{\min}|)\}_{j=1}^N$, thereby accounting for *both* road boundaries. For increased safety/larger $\Delta e_{y,\tau}$, Γ must be further decreased.

SUMMARY 11. *Two different methods were proposed to account for vehicle dimensions. In practice, a hybrid is proposed. Thus, the inflation and linearization approach are preferred for high and low-velocity applications, respectively.*

2.1.11 State Estimation and Environment Modeling

We assume there exists an area surrounding the ego car, referred to as "range field" $\mathcal{R}$ with length $l_{\mathcal{R}}$ at front, and in which objects are detectable, see Figure 15. Only for proof of concept it is modeled as rectangular. The range field may be time-varying. It is typically realized through lidar systems [167], [337]. Additionally, *car-2-car* communication can extend the visibility of otherwise shielded vehicles, and be employed for higher-level services [93] and multi-vehicle coordination [154].

Extended Kalman Filter

We define as *proprioceptive measurements* ($\mathcal{T}_{\text{V,E}}$) all of a subset of the vehicle state vector. As *exteroceptive measurements* ($\mathcal{T}_{\text{X,E}}$), we define any static or mobile objects within the range field, denoting the information retrieved about the objects at sampling time kT_s by $\eta_k^{t,i}, \forall i = 1, \ldots, N^{\text{obj}}$, whereby N^{obj} is the number of identified objects. We set

$$\eta_k^{t,i} = \left[d_k^{\text{obj},i},\ e_{y,k}^{\text{obj},h,i},\ e_{y,k}^{\text{obj},l,i},\ l_k^{\text{obj},i},\ v_{x,k}^{\text{obj},i} \right]^{\mathsf{T}}, \tag{2.38}$$

fitting all objects as rectangles with particular length and width aligned with the road at the road-projected object position, see Figure 15. This environment modeling approach is tailored to our control design. Vector $\eta_k^{t,i}$ represents the starting state from which the movement of the object can be predicted over a spatial horizon. The object lateral displacements w.r.t. road centerline, $e_{y,k}^{\text{obj},l,i}$ and $e_{y,k}^{\text{obj},h,i}$, and the object length $l_k^{\text{obj},i}$ allow to formulate constraints on e_y. As the database information ($\mathcal{T}_{\text{M,E}}$), we assume the following road information vector

$$r_i = [x_{s,i}, y_{s,i}, \psi_{s,i}, \psi'_{s,i}, \rho_{s,i}, v_{x,i}^{\text{road}}, e_{y,i}^{\text{road,max}}, e_{y,i}^{\text{road,min}}, s_i]$$

to be available along the road centerline at samples $i = 0, 1, \ldots, N^{\text{road}}$, where $v_{x,i}^{\text{road}}$ denotes the speed limits and $e_{y,i}^{\text{road,max}}$ and $e_{y,i}^{\text{road,min}}$ indicate the lane width along the track.

Model-based recursive estimation techniques are required for sensor fusion and reconstruction of system states in the presence of model un-

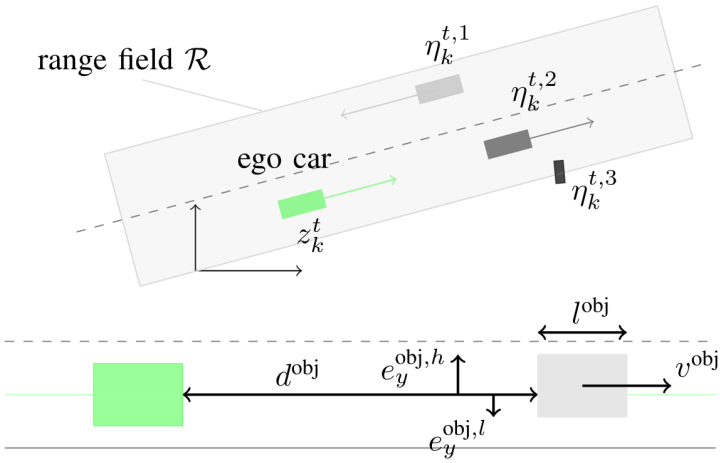


Figure 15: Top: measurement model (proprioceptive and exteroceptive). Within a particular "range field" $\mathcal{R}$ surrounding the ego car, objects are assumed to be detectable. Bottom: illustration of the exteroceptive measurement model with five states. The green and gray areas denote the ego and a leading car, respectively.

certainty and unmeasured states. A nonlinear recursive filtering approach is the *Extended Kalman Filter* (EKF), see [344], [9], [377]. It is often suitable for approximately Gaussian, i.e., unimodal noise. Due to the nonlinear vehicle dynamics, we employ a *discrete* EKF (dEKF) to compute an estimate $\hat{z}_k^t$ of z_k^t. Note that for the estimator design we linearize and discretize using the *time*-dynamics, see (2.12). This is in contrast to the control design, where the spatial-based system equivalent is employed. For exteroceptive measurements, we similarly design one Kalman Filter per object, $i = 1, \ldots, N^{\text{obj}}$, to obtain $\hat{\eta}_k^{t,i}$. To deal with situations in which detection of an object is temporarily lost, e.g., due to turning or shadow cones, we ensure that an object estimate is still returned using now only the prior update of the Kalman Filter equations (not the model-correcting measurement update). Only after a certain number of consecutive steps without new measurement, the object is dismissed and the Kalman filter reinitialized. In [155], dEKFs are *always* active. Thus, they process any obtained measurements, even if these are noise-free.

Conversion from Time to Space Domain

The time-based state estimate $\hat{z}_k^t$ returned by the dEKF is converted to the spatial-based $\hat{z}_k$ and $\hat{s}_k$ by using the road information. For the dynamic vehicle model case from [155], all fixed-body estimates of vehicle states $v_{x,k}$, $v_{y,k}$, ω_k, $\omega_{f,k}$ and $\omega_{r,k}$, are the same in both domains. For computation of $\hat{e}_{\psi,k}$, $\hat{e}_{y,k}$, we first project the current ego car position to the road centerline (projection of a point-mass to a piecewise-affine line). Then, we determine $\hat{e}_{\psi,k} = \hat{\psi}_k^t - \hat{\psi}_s^{\text{proj}}$ and $\hat{e}_{y,k} = c\sqrt{(\hat{x}_k^t - \hat{x}_s^{\text{proj}})^2 + (\hat{y}_k^t - \hat{y}_s^{\text{proj}})^2}$, where $\hat{x}_s^{\text{proj}}$, $\hat{y}_s^{\text{proj}}$ and $\hat{\psi}_s^{\text{proj}}$ indicate the position and orientation of the road centerline at the vehicle's projection point, and where the sign of $\hat{e}_{y,k}$ is specified by $\theta = \tan^{-1}(\frac{\hat{y}_k^t - \hat{y}_s^{\text{proj}}}{\hat{x}_k^t - \hat{x}_s^{\text{proj}}})$, and $c = -1$ if $\theta - \hat{\psi}_s^{\text{proj}} \in (0, -\pi]$, or $c = 1$ otherwise.

2.1.12 Road Navigation

Two different approaches are considered for road navigation. For both approaches, dynamic and kinematic vehicle models can be employed, and a driving corridor as in (2.35) is assumed. Then, the two approaches differ mainly in their objective function formulation.

The first approach from [155] involves the tracking error between states and a reference trajectory, i.e., $z_j - z_j^{\text{ref}},\ \forall j = 1, \dots, N$. The reference may, for example, represent the road centerline trajectory (including potential velocity and rate references). It may also represent a reference computed from a reference generator, for example, when overtaking or navigating among obstacles. Then, the objective function of the LSV-MPC formulation assumes the form

$$\begin{aligned}\min \ & \sum_{j=1}^{N-1} \|z_j - z_j^{\text{ref}}\|_{Q_z}^2 + \|z_N - z_N^{\text{ref}}\|_{Q_{zN}}^2 + \text{slack variables} + \\ & \sum_{j=0}^{N-1} \|u_j - u_j^{\text{ref}}\|_{Q_u}^2 + \|u_j - u_{j-1}\|_{Q_{\Delta u}}^2 + \text{other terms},\end{aligned} \tag{2.39}$$

where, in general, also control reference tracking errors and control rates are penalized, and the notation $\|z\|_Q^2 \triangleq z^\mathsf{T} Q z$ is here used for $z \in \mathbb{R}^n$ and a positive definite matrix $Q \in \mathbb{S}_{++}^n$. Thus, this approach characteristically requires to select hyperparameter matrices Q_z, Q_{zN}, Q_u and $Q_{\Delta u}$.

The second approach [148] does not explicitly involve a tracking error. Instead, the LSV-MPC objective function formulation reads

$$\min \ t_N + \max_{\{\delta_j\}_{j=0}^{N-1}} |\delta_j| + \max_{\{\delta_j\}_{j=0}^{N-2}} |\delta_{j+1} - \delta_j| \ + \text{slack variables} \tag{2.40a}$$

$$\text{s.t. } t_j^{\text{WP}} - \sigma \le t_j \le t_j^{\text{WP}} + \sigma,\ z_j \in \mathcal{Z}_j^{\text{WP}},\ \forall j \in \mathcal{J}^{\text{WP}}, \tag{2.40b}$$

$$\text{and other constraints},$$

where $\{\delta_j\}_{j=0}^{N-1}$ denote steering angles over the prediction horizon N. The absolute value is denoted by $|\cdot|$. The objective function (2.40a) trades-off time optimality and a minmax-type objective resulting in minimized steering actuation (*smooth steering*). Note that this objective results in a

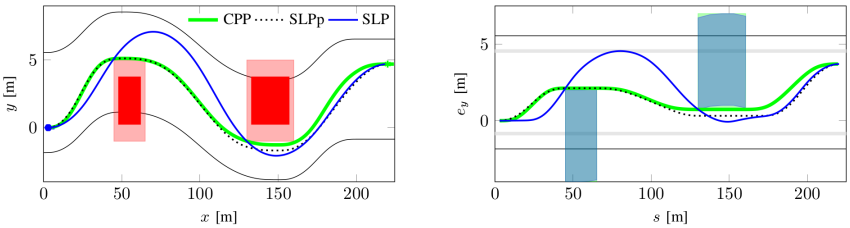


Figure 16: Example from [156]. Comparison of the resulting trajectories in the global and in the road-aligned plane, in which computations are conducted. The obstacles (red) are inflated by a safety margin (light red). The light gray lines (right plot) indicate road bounds $e_y^{\min}(s)$ and $e_y^{\max}(s)$. SLP: sequential linear programming, SLPp: as SLP but with overtaking in parallel (in the (s, ey)-frame) according to (2.41), and CPP: clothoid-based path planning (concatenating clothoid segments).

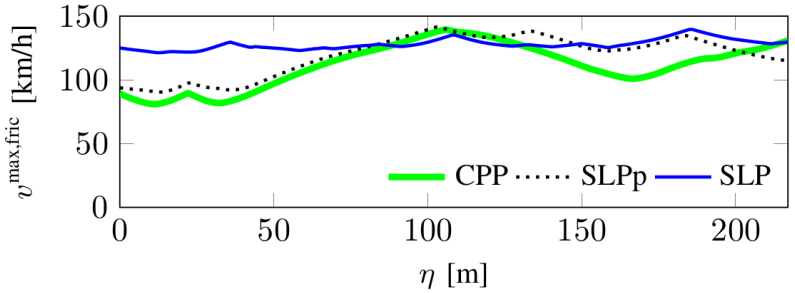


Figure 17: Example from [156]. The maximum admissible traveling speed within vehicle tire friction limits is overall significantly higher for SLP in comparison to CPP and SLPp. For SLP, the lowest $v^{\max,\text{fric}}(\eta)$ along its traveled path coordinate η is 121km/h. For CPP, the equivalent is 81km/h. Thus, the road segment could in principle be traversed much faster for SLP while still remaining in the safe vehicle tire friction domain.

linear program (LP) when reformulating the two minmax objectives after introduction of two nonnegative scalar slack variables. Spatiotemporal constraints (2.40b) are used for time scheduling. They indicate the times at which *waypoints* (WP) are meant to be traversed. We define $\mathcal{J}^{\text{WP}} = \{j : s_j = s_j^{\text{WP}},\ s_j^{\text{WP}} \in \mathcal{S}^{\text{WP}},\ j = 1, \dots, N\}$, where $\mathcal{S}^{\text{WP}}$ is an input set that may be provided, for example, by a higher-level mission planning algorithm. In addition to $\mathcal{S}^{\text{WP}}$, such algorithm must provide the corresponding scheduling times $\mathcal{T}^{\text{WP}} = \{t_j^{\text{WP}},\ \forall j \in \mathcal{J}^{\text{WP}}\}$. The states in which the waypoints are reached is constrainted by $\mathcal{Z}_j^{\text{WP}}$. To summarize, instead of an QP for reference tracking, in (2.40) a LP with minmax objective is formulated. Therefore, velocity is controlled via the formulation of spatiotemporal waypoints, and road navigation is achieved via minimized steering actuation within the provided driving corridor.

Additional constraints can easily be added. To enforce the overtaking of L obstacles in *parallel* (without specifying the lateral distance though), we may add

$$e_{\psi,l}^{\text{obs}} - \sigma_{e_\psi}^N \leq e_{\psi,j} \leq e_{\psi,l}^{\text{obs}} + \sigma_{e_\psi}^N,\ \forall j \in \mathcal{J}_l^{\text{obs}}, \tag{2.41}$$

for $l = 1, \dots, L$, with

$$\mathcal{J}_l^{\text{obs}} = \{j : s_l^{\text{obs,b}} \leq s_j \leq s_l^{\text{obs,e}},\ j = 1, \dots, N\},$$

and where $e_{\psi,l}^{\text{obs}}$ denotes the heading of the rectangle-envelope of obstacle l, located between $s_l^{\text{obs,b}}$ and $s_l^{\text{obs,e}}$. In [156], for constraint softening we reused slack variable $\sigma_{e_\psi}^N \geq 0$ (from a terminal state constraint on $e_{\psi,N}$) to not introduce a new decision variable.

Note that the LP-minmax approach increases safety. This is since minimized steering actuation implies lower path curvature, and thereby naturally also a higher admissible vehicle velocity that still permits vehicle operation within tire friction limits, see Figures 16 and 17.

For further clarification, the entire LP-minmax formulation labeled TOSS (time-optimal smooth steering) from [148] is stated:

$$\min\ t_N + \max_{\{\delta_j\}_{j=0}^{N-1}} |\delta_j| + \max_{\{\delta_j\}_{j=0}^{N-2}} |\delta_{j+1} - \delta_j| + W_\sigma \sum\nolimits_{i=1}^{4} \sigma_i \tag{2.42a}$$

$$\text{s.t. } z_0 = z(s_\tau),\ t_0 = \tau,\ u_{-1} = u(s_\tau - D_s) \tag{2.42b}$$

$$z_j = \begin{bmatrix} e_{\psi,j} & e_{y,j} \end{bmatrix}^\mathsf{T},\ j = 0, \dots, N, \tag{2.42c}$$

$$u_j = \begin{bmatrix} q_j & \delta_j \end{bmatrix}^\mathsf{T},\ j = 0, \dots, N-1, \tag{2.42d}$$

$$z_{j+1} = A_j z_j + B_j u_j + g_j,\ j = 0, \dots, N-1, \tag{2.42e}$$

$$t_{j+1} = t_j + \frac{D_{s,j}(\rho_{s,j}^{\text{ref}} - e_{y,j}^{\text{ref}})q_j}{\rho_{s,j}^{\text{ref}} \cos(e_{\psi,j}^{\text{ref}})},\ j = 0, \dots, N-1, \tag{2.42f}$$

$$e_\psi(s_\tau + S) - \sigma_1 \leq e_{\psi,N} \leq e_\psi(s_\tau + S) + \sigma_1, \tag{2.42g}$$

$$e_y(s_\tau + S) - \sigma_2 \leq e_{y,N} \leq e_y(s_\tau + S) + \sigma_2, \tag{2.42h}$$

$$e_{y,j}^{\min} + \Delta e_{y,\tau} - \sigma_3 \leq e_{y,j} \leq e_{y,j}^{\max} - \Delta e_{y,\tau} + \sigma_3, \quad j = 1, \dots, N, \tag{2.42i}$$

$$t_j^{\text{WP}} - \sigma_4 \leq t_j \leq t_j^{\text{WP}} + \sigma_4,\ z_j \in \mathcal{Z}_j^{\text{WP}},\ \forall j \in \mathcal{J}^{\text{WP}}, \tag{2.42j}$$

$$c_{q,j}^{\min} \leq b_{q,j+1} q_{j+1} + b_{q,j} q_j \leq c_{q,j}^{\max},\ j = 0, \dots, N-2, \tag{2.42k}$$

$$\tilde{T}\dot{\delta}^{\min} \leq \delta_{j+1} - \delta_j \leq \tilde{T}\dot{\delta}^{\max},\ j = 0, \dots, N-2, \tag{2.42l}$$

$$\frac{1}{v^{\max}} \leq q_j \leq \frac{1}{v^{\min}},\ j = 0, \dots, N-1, \tag{2.42m}$$

$$\delta^{\min} \leq \delta_j \leq \delta^{\max},\ j = 0, \dots, N-1, \tag{2.42n}$$

$$\sigma_1 \geq 0,\ \sigma_2 \geq 0,\ \sigma_3 \geq 0,\ \sigma_4 \geq 0, \tag{2.42o}$$

with decision variables $\{u_j\}_{j=0}^{N-1}$, $\{\sigma_i\}_{i=1}^{4}$ and optimization horizon N. The absolute value is denoted by $|\cdot|$. Objective function (2.42a) trades-off time-optimality and a minmax-type objective resulting in minimized steering actuation (*smooth steering*). Note that W_σ penalizes slack variables, therefore typically selected large (e.g., 10^4), and represents the only weight in (2.42a). Thus, in contrast to the QP-based approach from (2.39), the preferred LP-minmax approach does not require to select any tuning parameters. Hard constraints (2.42m) are derived from $v^{\min} \leq v_j \leq v^{\max},\ j = 0, \dots, N-1$, and from the coordinate transformation according to Section 2.1.7, whereby $v^{\max}$ denotes the road speed limit and $v^{\min}$ the minimum permissible velocity. Since (2.42e), (2.42f) and (2.42k) depend

on reference trajectories, (2.42) is solved *twice* according to Section 2.1.6. For the first iteration, we initialize state reference trajectories along the road centerline, i.e., $e_{\psi,j}^{\text{ref}} = 0$ and $e_{y,j}^{\text{ref}} = 0,\ \forall j$, and select $q_j^{\text{ref}} = \frac{1}{v_\tau}$ and $\delta_j^{\text{ref}} = 0,\ \forall j$. Finally, moving obstacles are accounted for by their *velocity*- and *trajectory*-adjusted mappings to the road-aligned coordinate frame according to the method of [155, Sect. III-E], which is also discussed in Section 2.1.14.

An example is provided in Figure 18. The minimum-time traversal of a curvy road segment with one obstacle is sought. Two scheduling constraints are considered: $\mathcal{S}^{\text{WP}} = \{s^{\text{obj}}, 170\}$ and $\mathcal{T}^{\text{WP}} = \{10, 16\}$, where s^{obj} denotes the coordinate at which the obstacle is first encountered. For both waypoints, we did not further constrain admissible lateral vehicle position. Note that only for better visualization of presented concepts both road lanes were admitted for maneuvering. In practice, the permissible road width can be controlled conveniently by spatially varying bounds on e_y. Several observations can be made. First, the spatiotemporal waypoints are met accurately. See Figure 18 for the resulting optimal velocity trajectory. Second, TOSS combines steering *and* velocity control.

SUMMARY 12. *The preferred LSV-MPC formulation involves a LP-minmax objective. Therefore, weighting matrices for explicit tracking of reference trajectories, which are typical for QP-based MPC, are not required. The minimized steering actuation objective in combination with lateral driving corridor constraints is likewise beneficial for increased safety. Minimized steering actuation implies lower path curvature, and thereby naturally also a higher admissible vehicle velocity that still permits vehicle operation within tire friction limits.*

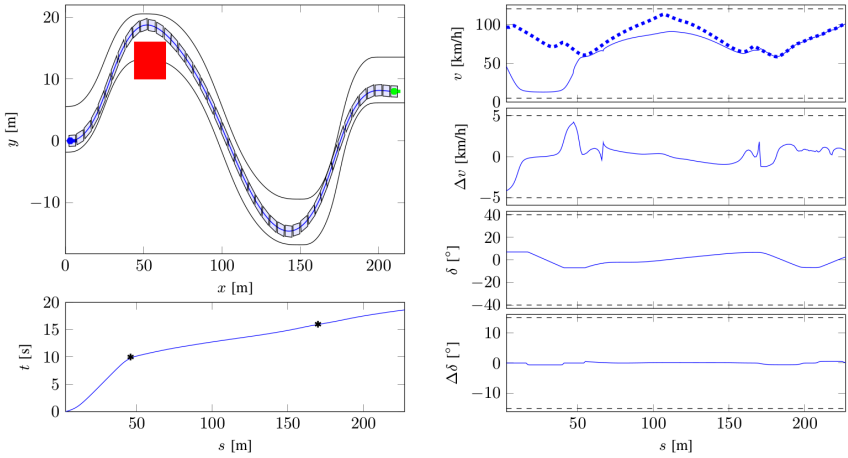


Figure 18: Example from [148]. State, time and control trajectories. The obstacle is indicated in red. The two scheduling times and corresponding spatial coordinates are visualized by two black asterisks. Control absolute and rate constraints are indicated by black dashed lines. Velocity $v^{\text{max,fric}}(s)$ is displayed by the blue dotted line. Solutions of TOSS are indicated by blue solid lines.

2.1.13 Zone Navigation

Road navigation can be considered as a constrained zone navigation task when treating road boundaries as "obstacles". The road centerline is naturally provided by the road profile. For *zone navigation* tasks the road centerline, which serves naturally as the reference during road following tasks, is absent. Therefore, in order to create such a reference, a sequential convex optimization approach is considered as follows:

1. Dismissing the obstacles at first, connect the vehicle start and end pose by a piecewise-affine (PWA) trajectory.
2. Use the PWA trajectory as a reference for a *time*-based LTV-MPC tracking controller.
3. Use the LTV-MPC trajectory as the "road centerline" in a *spatial*-based LSV-MPC framework, mapping obstacles, planning a corridor (see Section 2.1.14), and accounting for vehicle dimensions in a road-aligned coordinate frame (see Section 2.1.10).

Several remarks can be made. First, the heuristic for PWA trajectory generation is largely shaping the overall result. For example, one may alternatingly generate rays starting from the end and start vehicle pose, avoiding intersections in start and end positions to encourage outreaching steering behavior, and turning perpendicularly after, for example, 5m (or the minimum velocity-dependent turning radius) either always counterclockwise (CCW) or always clockwise (CW) if no intersection is yet reached. In Figure 19, the CW-solution is displayed. This heuristic can be extended by *intermediate locations* when admitting changes of velocity sign. Second, as demonstrated the method succesfully permits tight maneuvering. In general, a LTV-MPC tracking controller based on the nonlinear kinematic bicycle model [317] was found to be very robust even when tracking edgy paths, see also Figure 5. Since LTV-MPC is time-parameterized and thus heavily dependent on reference velocity, constraint formulation for obstacle avoidance is problematic as discussed in Section 2.1.3. This is the reason for the subsequent *spatial*-based LSV-MPC. Then, its trajectory output is used as reference for a *second*

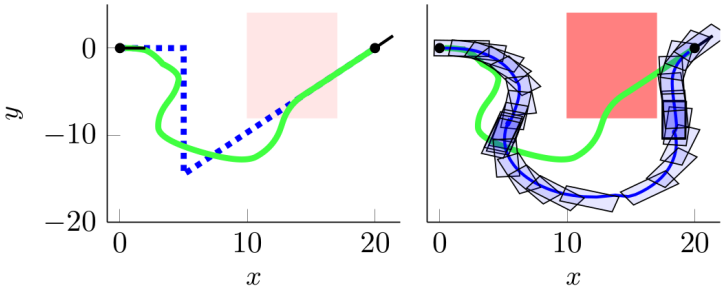


Figure 19: A freeform navigation task with one obstacle. The example is at the limits of what can be handled by iterative linearization in three steps. A trajectory plan is sought connecting the two black indicators, avoiding the red obstacle and accounting for vehicle dimensions. (Left) Dismissing the obstacle, the heuristic PWA trajectory (blue dotted) and LTV-MPC tracking result (green) are shown. (Right) The final trajectory after two iterations of spatial-based MPC with linearized vehicle dynamics. A hybrid of [156] and [148] is used with the time-optimal smooth-steering objective (2.42a). In contrast to *rectangle-envelopes* aligned with the road-centerline in [156], obstacle contours are mapped directly and without any margins. This enables maximally tight maneuvering.

spatial-based LSV-MPC, which generates a further smoothed trajectory, and permits to use friction constraints and also time-scheduling [148]. Despite tight obstacle avoidance for the example in Figure 19, several limitations are obvious. First, there are small wiggles (including a decrease in output velocity) in the final trajectory resulting from the very large deviations between LTV-MPC reference and final trajectory (after linerization and accounting for obstacles). Second, the dependence on reference trajectories is apparent. Third, the computational complexity associated with three sequential MPCs is high.

2.1.14 Combinatorial Obstacle Avoidance and Corridor Planning

An important task of the motion planning system is to ensure that the ego car travels inside the lane boundaries while safely avoiding all obstacles (*corridor planning*). The problem is non-convex due to the fact that overtaking is possible on both sides of obstacles. It is further complicated because of a time-varying environment (with dynamically moving objects) requiring possibly frequent recursive replanning of corridors. Therefore, in [155], the *combinatorial obstacle avoidance* problem about the decision on which side to overtake obstacles is handled by a variant of a *label correcting algorithm* (LCA) [47] using the cumulatively least-heading-varying path as optimization criterion. This method permits to formulate convex driving corridors required for the formulation of QP- or LP-based LSV-MPC problems.

At every sampling time along the corridor coordinate s, there needs to be a *velocity-* and *trajectory*-adjusted mapping of all objects. This adjustement is achieved by solving for every object the equation $s(t^\star) = s^{\text{obj}}(t^\star)$ for $t^\star$ and $s(t^\circ) = s^{\text{obj}}(t^\circ) + l^{\text{obj}}(t^\circ)$ for t°, where $s(t)$ is the ego car CoG-position along the road centerline of the corridor at time t and similarly for the object. The velocity-adjusted object positions are then located between $s(t^\star)$ and $s(t^\circ)$. Including additional safety margins, they are used for graph generation, see Algorithm 1. At the time instant of corridor planning, we model the progress of the ego car and other vehicles with *constant* velocities, namely the velocities at that time instant, and thus obtain $s(t) = \hat{s}_k + (t - kT_s)\hat{v}_{x,k}$, for $t \geq kT_s$, and similarly for the objects. Then, we can derive

$$s(t^\star) = \hat{s}_k + \left(\frac{\hat{s}_k^{\text{obj}} - \hat{s}_k}{\hat{v}_{x,k} - \hat{v}_{x,k}^{\text{obj}}} \right) \hat{v}_{x,k}, \tag{2.43}$$

$$s(t^\circ) = s(t^\star) + \frac{\hat{l}^{\text{obj}} \hat{v}_{x,k}}{\hat{v}_{x,k} - \hat{v}_{x,k}^{\text{obj}}}. \tag{2.44}$$

The lateral displacement w.r.t. road centerline can then be similarly evaluated as $e_y(t^\star)$ and $e_y(t^\circ)$.

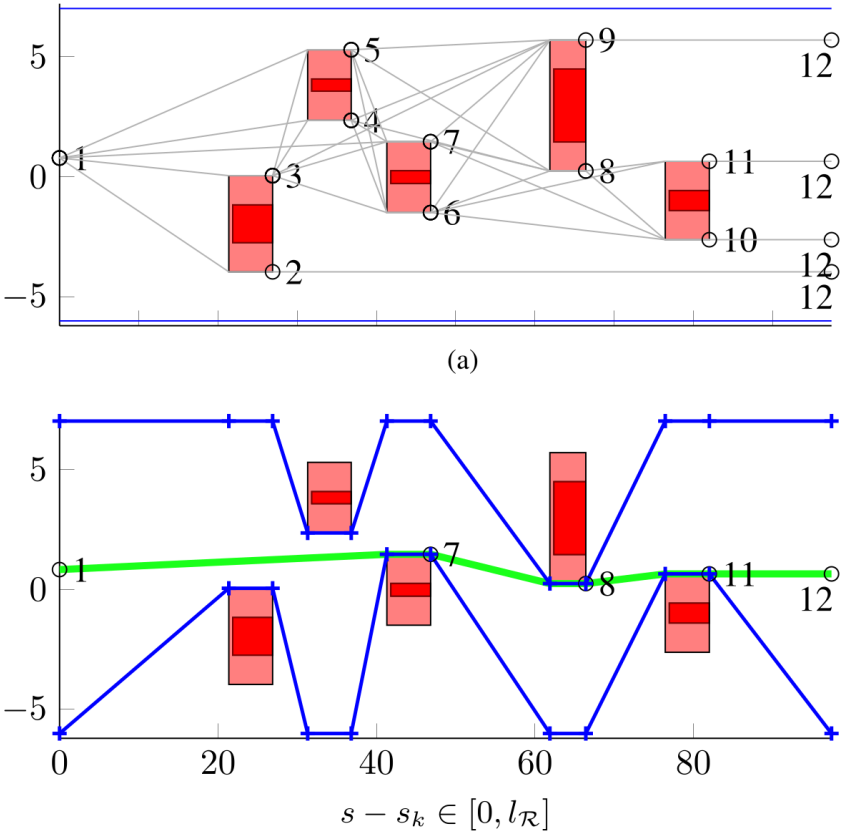


Figure 20: Illustration of the geometric corridor planner according to [155]. The red rectangles denote solid obstacles and the lighter red areas include safety margins. The green PWA line illustrates the *cumulatively least-heading-varying path* determined on the graph. This corridor planning scheme is summarized in Algorithm 2.

Algorithm 1: Graph creation for corridor planning

Preparation: given *velocity-* and *trajectory*-adjusted object positions within the corridor, fill $P_{\text{list}} \in \mathbb{R}^{(1+2N^{\text{obj}})\times 3}$ using the starting node and both boundary positions of each object.

Initialize: tree $T \in \mathbb{R}^{(2+2N^{\text{obj}})\times(2+2N^{\text{obj}})}$ with $T_{j_t j_t} \leftarrow 0$ where $j_t = (2 + 2N^{\text{obj}})$ is the terminal node ($s = l_{\mathcal{R}}$ but potentially variable e_y-positions), and list OPEN $\leftarrow \{1\}$.

while *list OPEN is not empty:* **do**

  Remove a node i from OPEN and determine the corresponding s_i and $e_{y,i}$ from P_{list}.

  if *there exists a direct path from* s_i *to* $s = l_{\mathcal{R}}$ *on* $e_y = e_{y,i}$ *without intersecting any other object* **then**

    Set $T_{ij_t} \leftarrow 0$.

  else

    Identify all objects ahead with $s > s_i$.

    for *each object in front* **do**

      Find corresponding two object boundary nodes j_1 ('left' of object) and j_2 ('right' of object).

      if *transition from node* i *to* j_1 *is not intersecting with any other object* **then**

        Compute $\Delta\theta_{ij_1}$, see (2.45).

        if $\Delta\theta_{ij_1} < \Delta\theta^{max}$ **then**

          $T_{ij_1} \leftarrow \Delta\theta_{ij_1}$.

        if $s_{j_1} + l_{j_1} \geq l_{\mathcal{R}}$ **then**

          $T_{j_1 j_t} \leftarrow 0$.

        else

          Add node j_1 to OPEN if it is not already in OPEN.

      Do the same operations for the transition from node i to j_2 as for from i to j_1.

Remove any nodes that were never reached from T and P_{list}.

Algorithm 2: Corridor planning

Input: corridor area, velocity- and safety margin-adjusted obstacle locations, initial ego car position and heading.
Order objects according to increasing s-position along the corridor and identify corresponding road bounds.
Dismiss objects significantly faster and in front or slower and behind the ego car position.
Adjust road bounds if objects overlap longitudinally with ego car CoG-position, or a neighboring lane is blocked.
if *there still remain objects within the updated corridor:* **then**
    Build a transition graph T and store the corresponding node positions in P_{list} using Algorithm 1.
    Use the Label Correcting Algorithm (LCA) [47] to find the cumulatively least-heading-varying path $\bar{e}_y^{\text{ref}}$.
    Given $\bar{e}_y^{\text{ref}}$ and the object locations, determine $e_y^{\min}$ and $e_y^{\max}$. See Figure 20 for illustration.
else
    Set $e_y^{\min}$ and $e_y^{\max}$ to the adjusted road bounds, set $\bar{e}_y^{\text{ref}}$ to the e_y-level of a designated road lane .

We develop a path planner using graph theory for computational efficiency. To address the issue of trajectory feasibility, we follow the approach of heuristically augmenting objects by velocity-dependent safety margins chosen according to closed-loop driving simulation over the velocity range 30-130km/h. We select lateral safety margins as a linear function of $\hat{v}_{x,k}$ such that at $\hat{v}_{x,k} = 130$km/h a safety distance of at least 2m between the edge of the ego car and the boundary of another object is maintained. Selections for front and rear longitudinal safety distances are similarly linearly dependent on $\hat{v}_{x,k}$ and $\hat{v}_{x,k}^{\text{obj}}$. We further encourage an early start of steering on sight of an obstacle, thereby avoiding large incremental steering changes. Fundamental is the reachability of areas lateral of obstacles free for trespassing. Following the projection point of such areas ensures (geometrically) minimal heading variation. Possible trajectories can conveniently be modeled as a transition graph, with the absolute heading variations as edge weights. The trajectory that provides *cumulated minimal absolute heading variation* and safe avoidance of all ob-

stacles can be considered for the adjustment of bounds on e_y. The complexity of the corridor planning problem increases with the number of obstacles. Assuming N^{obj} objects longitudinally displaced along a corridor, there are $2^{N^{\text{obj}}}$ possible combinations of overtaking. Our method naturally also allows for longitudinally overlapping (but laterally displaced) object constellations. It is outlined in Algorithm 2. For the creation of $e_y^{\min}$ and $e_y^{\max}$ in Step 8 of Algorithm 2 there exist different methods such as stairwise or smoothed adaptation. Our preferred method is indicated in Figure 20 making use of the s-coordinates of all objects. In Algorithm 2, the graph generation step is computationally and conceptually decisive. Modeling objects as rectangles aligned with the road centerline allows one to transition between nodes via two piecewise-affine (PWA) lines (constant slope and constant e_y-level). See Figure 20 for illustration. For N_{nodes} different nodes, each node position $i \in \{1, \dots, N_{\text{nodes}}\}$ is summarized in P_{list} by $P_{\text{list},i} = \begin{bmatrix} s_i & l_i & e_{y,i} \end{bmatrix}$, where s_i denotes the s-position where the constant e_y-level line-part begins, l_i the length of this part and $e_{y,i}$ the corresponding e_y-level. For node 1 and the terminal node we set $l = 0$. Then, a transition matrix $T \in \mathbb{R}^{N_{\text{nodes}} \times N_{\text{nodes}}}$ can be defined with

$$T_{ij} = \begin{cases} \Delta\theta_{ij}, & \text{if } \exists \text{ a PWA transition } i \to j \\ \infty, & \text{otherwise,} \end{cases}$$
$$\Delta\theta_{ij} = \left| c - \tan^{-1}\left(\frac{e_{y,j} - e_{y,i}}{s_j - (s_i + l_i)} \right) \right| \tag{2.45}$$

where $c = \hat{e}_{\psi,k}$ for $i = 1$ and $c = 0$ for $i \in \{2, \dots, N_{\text{nodes}}\}$.

A characteristic behavior of the devised path planner is to start steering early, i.e., "on sight" of an obstacle. This is beneficial since it results in minimal incremental steering changes, while still allowing for a quick heading correction if required. Importantly, a timely OA-maneuver initiation provides an early (partial) insight into a neighboring lane and thereby may allow for detection of previously shielded objects if there is no car-2-car communication, or, in case of car-2-car communication, visual confirmation of the communicating vehicles. The PWA reference trajectories returned by the corridor planner are either directly fed to the spatial-based predictive controller, or may first additionally be smoothed.

2.1.15 Adaptive Cruise Control

Besides obstacle avoidance and road tracking capability, *adaptive cruise control* (ACC) capability is the third fundamental requirement for road navigation. In [155], we introduce a simple discrete-time difference equation to model the longitudinal distance between the controlled vehicle and a leading object, i.e.,

$$d_{l+1}^{\text{obj}} = d_l^{\text{obj}} + T_s \left(v_l^{\text{obj}} - v_{x,l} \right), \tag{2.46}$$

where v_l^{obj} is the object velocity at time $t = lT_s$, $l > 0$, and the sampling time T_s is potentially dependent on velocity $v_{x,l}$. The distance dynamics (2.46) can be converted to the space domain by substitution such that

$$d_{j+1}^{\text{obj}} = d_j^{\text{obj}} + \frac{D_s(\rho_{s,j} - e_{y,j}) \left(v_j^{\text{obj}} - v_{x,j} \right)}{\rho_{s,j}(v_{x,j}\cos(e_{\psi,j}) - v_{y,j}\sin(e_{\psi,j}))} \tag{2.47}$$

is obtained when assuming a uniform discretization spacing D_s. Notice that, in contrast to the time domain, the dynamics of the longitudinal distance in (2.47) are nonlinear in the space domain. Furthermore, for a given a starting distance w.r.t. a leading object, we can write out the recursion of (2.47) as just a function of the automated vehicle's states including $e_{y,j}$, $v_{x,j}$, $e_{\psi,j}$ and $v_{y,j}$. Then, the discretized model reads

$$d_{j+1}^{\text{obj}} = d_j^{\text{obj}} + a_{d,j} z_j + g_{d,j},$$

where $d_j^{\text{obj}} \in \mathbb{R}$, $a_{d,j} \in \mathbb{R}^{1 \times n_z}$, $g_{d,j} \in \mathbb{R}$ and $j \in \mathbb{N}$ indexes steps over distance $s \in [jD_s, (j+1)D_s]$.

Then, for the ACC-driving mode, the LSV-MPC formulation can read

$$\min \sum_{j=2}^{N-1} \|d_j - d_j^{\text{ref}}\|_{q_d}^2 + \|d_N - d_N^{\text{ref}}\|_{q_{dN}}^2 + \text{and other terms} \tag{2.48a}$$

$$\text{s.t. } d_0 = \hat{d}_k^{\text{obj},i^*}, \text{ where } i^* \in \{1, \ldots, N^{\text{obj}}\}, \tag{2.48b}$$

$$d_{j+1} = d_j + a_{d,j} z_j + g_{d,j},\ j = 0, \ldots, N-1, \tag{2.48c}$$

$$d_j \geq d_{\min} + c_{\min} v_{x,j} - \sigma_d,\ j = 1, \ldots, N, \tag{2.48d}$$

$$\sigma_d \geq 0, \tag{2.48e}$$

$$\text{and other constraints}, \tag{2.48f}$$

where the velocity-dependent reference distance is defined as

$$d_j^{\text{ref}} = p_{\text{dref}} \hat{v}_{x,k} + d_{\min},\ j = 2, \ldots, N, \tag{2.49}$$

where $i^\star$ indicates the ACC-reference object (of N^{obj} many, including unsuitable static ones) with initial (estimated or measured) distance $\hat{d}_k^{\text{obj},i^\star}$, where $d_{\min}$ and $c_{\min}$ represent safety parameters, and where $d_{\min}$ and p_{dref} are calibrated. For example, $d_{\min} = 2$m and $p_{\text{dref}} = (27 - d_{\min})/(50/3.6)$ enforces a safety distance of 27m at 50km/h vehicle speed. This corresponds to the rule of thumb to keep a safety distance in meters of "slightly more than half of the speedometer indication in km/h". In order to not neceessarily accelerate the ego car to achieve this distance, we set $d_j^{\text{ref}} = 0.95\hat{d}_j^{\text{obj},i^\star} + 0.05 d_j^{\text{ref}}$ if $d_j^{\text{ref}} < \hat{d}_j^{\text{obj},i^\star}$. This safety-oriented strategy is appropriate for autonomous driving. In contrast, for cooperative driving (with enhanced predictability of neighboring communicating cars) we may use (2.49) directly as is in order to very quickly achieve platoonning [154].

Alternatively, and to maintain the preferred LP-form of (2.40), the reference tracking objective (2.48a) is dismissed and spatiotemporal waypoints are introduced instead. Therefore, $t_j^{\text{WP}} = \frac{d_j^{\text{ref}}}{v_j^{\text{ref}}}$ and $s_j^{\text{WP}} = d_j^{\text{ref}}$ may be set in (2.40b) and continuously updated online, where v_j^{ref} denotes the reference velocity (e.g., the minimum of road speed limit and lead obstacle velocity).

2.1.16 Driving Mode Selection Heuristics

The objectives of adaptive cruise control (with distance-keeping capabilities) and automated obstacle avoidance (approaching and overtaking of a leading object) are by definition conflicting. Therefore, it *must* be distinguished between at least these two driving modes. In [155], it is distinguished between four driving modes: adaptive cruise control (ACC), obstacle avoidance (OA), object-free road tracking (RT) and controlled braking (Brake).

For ACC-mode activation, we make use of knowledge about the ego car's braking capabilities. We assume that, for an ego car with velocity $v_{x,0}$ at position s_0 and for a given road surface and profile, a maximal possible brake deceleration $a_b^{\max}$ can be determined. In case of a constant deceleration, the position of the car after a time interval t can be computed as $s_t = a_b^{\max}\frac{t^2}{2} + v_{x,0}t + s_0$. We further assume a leading object with constant velocity v_0^{obj} and starting position s_0^{obj}, so that its position after time t is $s_t^{\text{obj}} = s_0^{\text{obj}} + v_0^{\text{obj}}t$. For crash avoidance in the interval $[0, t]$, it has to hold $s_t < s_t^{\text{obj}} - d_{\min}$, where $d_{\min}$ is a safety minimum distance. The time interval of interest is defined by N, the number of prediction steps N, i.e., $t = NT_s$. Defining $d_0^{\text{obj}} = s_0^{\text{obj}} - s_0$, we can compute the crash-critical initial velocity of the ego car as $v_{x,0} = \frac{1}{NT_s}(d_0^{\text{obj}} + v_0^{\text{obj}}NT_s - \frac{a_b^{\max}}{2}(NT_s)^2 - d_{\min})$, resulting in a final position $s_{NT_s} = s_{NT_s}^{\text{obj}} - d_{\min}$. Introducing an arbitrary safety factor $c \in (0, 1)$ to trigger steering at a lower velocity, the criterion for switching to OA-mode is thus to check at time kT_s if $\hat{v}_{x,k} > \hat{v}_{x,k}^{\text{crit},i^*}$ holds, with $\hat{v}_{x,k}^{\text{crit},i^*} = \frac{1}{NT_s}\left(c\hat{d}_k^{\text{obj},i^*} + \hat{v}_{x,k}^{\text{obj},i^*}NT_s - \frac{a_b^{\max}}{2}(NT_s)^2 - d_{\min}\right)$, where $a_b^{\max}$ is a function of $\hat{v}_{x,k}$ and $\hat{s}_k$. The critical velocity can be refined by taking into account detailed knowledge about braking dynamics, model-based probabilistic predictions of leading vehicle trajectories or inter-vehicular communicated braking trajectories. In all cases, $\hat{v}_{x,k}^{\text{crit},i^*}$ is determined from solving $s_{NT_s} = s_{NT_s}^{\text{obj}} - d_{\min}$.

A second consideration is to overtake a leading object whose velocity is too far off the road reference velocity. Naturally, this is allowable only in case of an available object-free neighboring lane permitting an obstacle

Algorithm 3: Driving mode selection (ACC, OA, RT, Brake)

Input: range field $\mathcal{R}$, velocity- and safety margin-adjusted location information of all N^{obj} obstacles in $\mathcal{R}$.
Order objects according to increasing s-position along the corridor and identify corresponding road bounds.
Dismiss objects significantly faster and in front or slower and behind the ego car position.
Group objects longitudinally and laterally overlapping.
Find an object of interest $i^\star \in \{1, \dots, N^{\text{obj}}\}$ with non-zero velocity (usually the closest to the ego car) for ACC.
if *there exists a neighboring lane which allows an obstacle avoidance maneuver* **then**
    if $(\hat{d}_k^{obj,i^*} > d_{min})$ && $(\hat{v}_{x,k}^{obj,i^*} \leq v_{x,k}^{road})$ && $(\hat{v}_{x,k}^{obj,i^*} > 0)$ && $(\hat{v}_{x,k} < \hat{v}_{x,k}^{crit,i^*})$ && $(\hat{v}_{x,k}^{obj,i^*} > v_{x,k}^{obj,min})$ **then**
        ACC;
    else if *there are objects within* $\mathcal{R}$ **then**
        OA;
    else if *there are no objects left after Step 3* **then**
        RT;
else
    if $(\hat{d}_k^{obj,i^*} > d_{min})$ && $(\hat{v}_{x,k}^{obj,i^*} \leq v_{x,k}^{road})$ && $(\hat{v}_{x,k}^{obj,i^*} > 0)$ && $(\hat{v}_{x,k} < \hat{v}_{x,k}^{crit,i^*})$ **then**
        ACC;
    else if *there still are objects within* $\mathcal{R}$ **then**
        Brake;
    else if *there are no objects left after Step 3* **then**
        RT;

avoidance maneuver. A possible heuristic for the minimal object velocity is $v_{x,k}^{\text{obj,min}} = v_{x,k}^{\text{road}}(0.4 + \max(0, \frac{v_{x,k}^{\text{road}}-30}{130-30}0.5))$.

A basic driving mode selection logic is described by Algorithm 3. We remark that the ACC-mode may be active while simultaneously performing obstacle avoidance of, e.g., static or very slow objects. A neighboring lane may not only be occupied due to an advancing vehicle but also because of a car approaching quickly from behind (a typical highway driving situation). Ultimately, an alternative triggering method may be based on a slack variable exceeding the minimum distance constraint. However, this is guaranteed to always trigger one sampling time later than object detection since a MPC problem has to be solved first to ouput the slack variable. In simulations this short delay could already render a succesful OA-maneuver impossible, eventhough it was feasible using the method described before (acting directly upon first object detection).

Simulation Experiments

In [155], simulation experiments are discussed that illustrate activations of the different driving modes. Figure 21 and 22 visualize two simulation experiments. For both, all four driving modes are activated for resolving the given driving scenarios. The average computation times are summarized in Table 1. Several remarks can be made. First, the effect of speed-dependent discretization spacing can be observed from the different N. Second, the computational burden for dEKF and geometric corridor planning is much less than for QP-building and solving. Third, note that the computation time required for QP-building is not negligible. Both building- and solve-time increase with increasing problem complexity and increasing N.

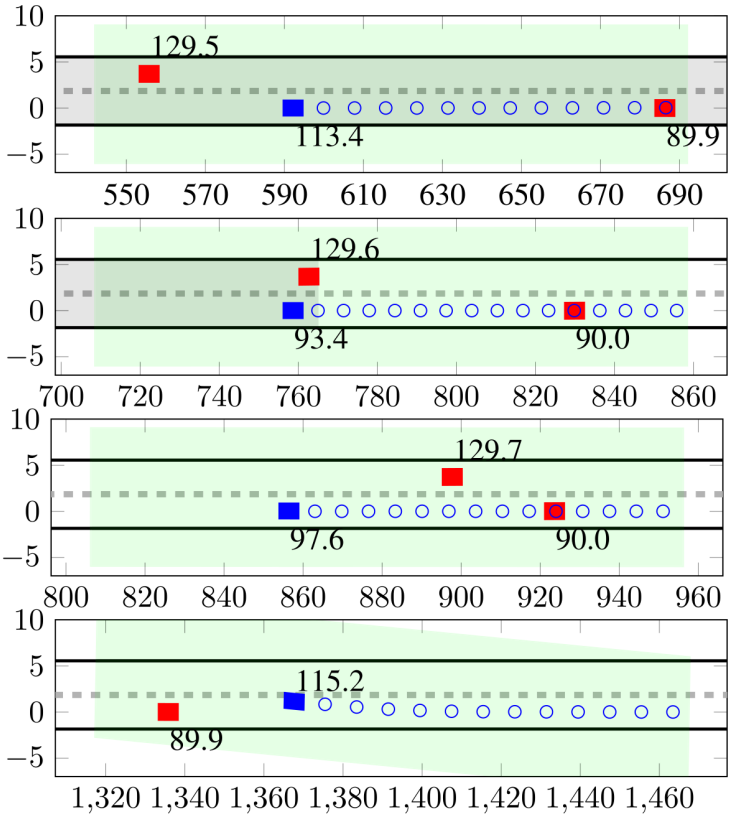


Figure 21: Highway simulation from [155]. The velocities [km/h] of the objects (red) and the ego car (blue) are indicated next to them. The blue dots indicate the reference trajectory of the ego car along the prediction horizon. The green area shows the range-field in which objects can be detected. The x- and y-axis indicate position coordinates in meters. (a–top frame) Due to a fast vehicle advancing from behind, the controller prohibits a lange change ('blocks' the left lane). Simultaneously, an object with slower velocity is detected in front and the braking-mode is triggered. (b) The ego car has decelerated its velocity sufficiently, and is in ACC-mode, adjusting its velocity to the leading vehicle. (c) The faster car has passed the ego car, the left lane is free for overtaking, and an obstacle avoidance maneuver (OA-mode) is initiated in order for the ego car to pursue its set reference velocity of 115km/h. (d) The ego car has overtaken the slower vehicle and is returning to road-profile tracking (RT-mode).

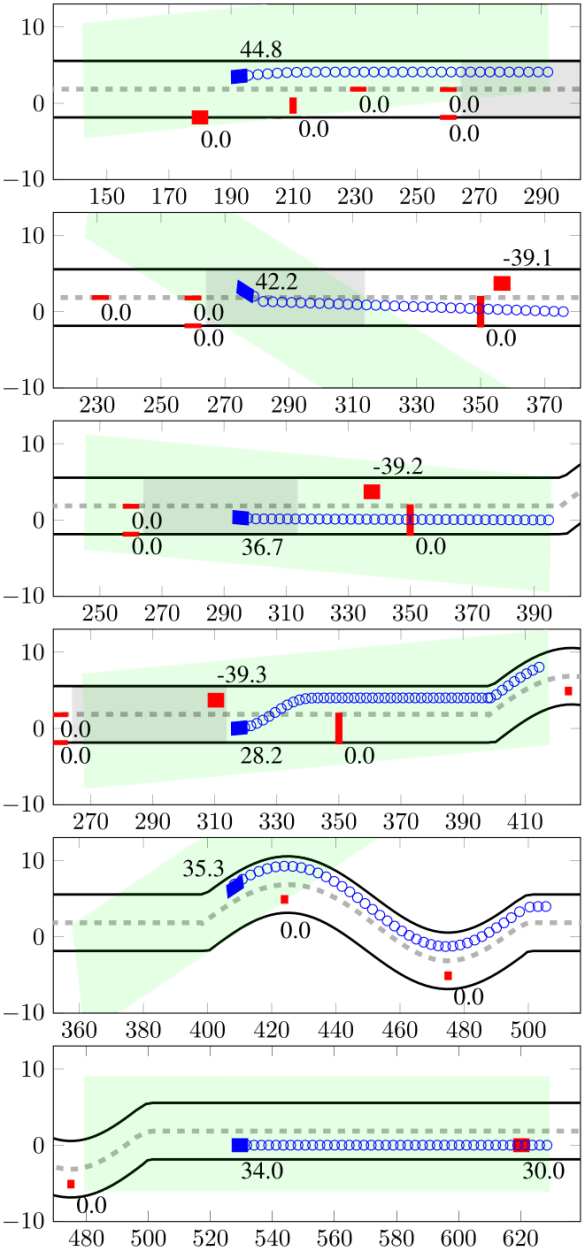


Figure 22: Multiple obstacles simulation from [155].

Table 1: Average computation times $\bar{\tau}$ in milliseconds for the simulation experiments in Figure 21 and 22. Time $\bar{\tau}_{\text{qp,build}}$ includes linearization, discretization and building of the quadratic program (via the elimination of states). Computation times using MATLAB's `quadprog` for the solution of the QPs are denoted by $\bar{\tau}_{\text{quadprog}}$. For the graph creation, see Algorithm 1, and for the subsequent finding of a cumulatively-least-heading varying path on the tree via LCA [47], $\bar{\tau}_{\text{tree,build}}$ and $\bar{\tau}_{\text{LCA}}$ resulted. The state estimator time is $\bar{\tau}_{\text{dEKF}}$. The average (velocity-dependent) prediction horizon is $\bar{N}$.

Property	Figure 21	Figure 22
$\bar{\tau}_{\text{dEKF}}$	1.1	1.1
$\bar{\tau}_{\text{tree,build}}$	0.7	1.0
$\bar{\tau}_{\text{LCA}}$	0.2	0.2
$\bar{\tau}_{\text{qp,build}}$	5.1	17.0
$\bar{\tau}_{\text{quadprog}}$	14.5	85.0
$\bar{N}$	12	38

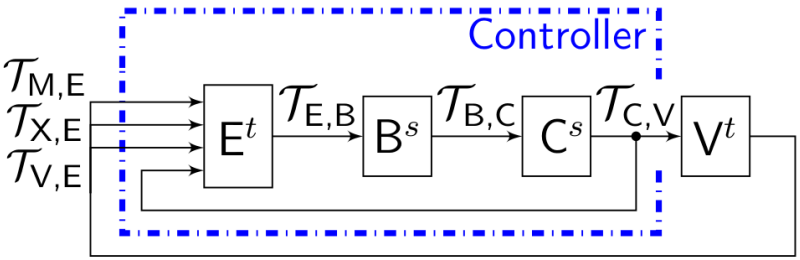


Figure 23: Hierarchical controller parametrization for single-vehicle motion planning by MPC. E^t (time-domain estimator): sensor fusion/model-based state estimation, adaptive driver model and perception/surrounding modeling. B^s (spatial-based builder): conversion from time- to space-domain, corridor planning, reference signal and constraints generation as well as driving mode selection. C^s (spatial-based controller): online LSV-MPC formulation and QP-solver. V^t (time-based vehicle model): low-level controllers and nonlinear vehicle dynamics. $\mathcal{T}_{\text{M,E}}$: maps and offline-generated database. $\mathcal{T}_{\text{X,E}}$: exteroceptive measurements, i.e., surrounding perception. $\mathcal{T}_{\text{V,E}}$: proprioceptive measurements. $\mathcal{T}_{\text{E,B}}$: perception and localization information. $\mathcal{T}_{\text{B,C}}$: constraint, reference and mode selection information. $\mathcal{T}_{\text{C,V}}$: high-level controls.

2.1.17 Hierarchical Controller Parametrization

For single-vehicle motion planning by MPC, the preferred control paramterization consists of three hierarchies as visualized in Figure 23. The three hierarchies are:

1. State estimation and environment modeling;
2. Corridor planning and reference trajectory design;
3. LSV-MPC optimization problem building and solution.

As illustrated in the simulation experiments from Section 2.1.16, the computational burden is by far the largest for the LSV-MPC layer. This layer includes a) the building and b) the solution of corresponding QPs or LPs. In Section 2.1.1 it was motivated that a space-*varying* LSV-MPC formulation (instead of a space-*invariant* LSI-MPC) is required for general motion planning. Therefore, the *online* building and solving of QPs or LPs cannot be avoided.

2.2 Single-vehicle Motion Planning by Neural Networks

This section summarizes [142]:

- M. Graf Plessen, "Automating vehicles by deep reinforcement learning using task separation with hill climbing," *arXiv preprint arXiv: 1711.10785*, 2017. (Submitted).

A model-based reinforcement learning (RL) method is proposed for the training of feedforward controllers in the context of autonomous driving. Fundamental philosophy is to offline train on arbitrarily sophisticated nonlinear models, while online cheaply evaluate a feedforward controller, thereby avoiding the need for online optimization. The contributions are, first, the discussion of two closed-loop control architectures, and, second, the proposition of a simple gradient-free algorithm for deep reinforcement learning using task separation with hill climbing (TSHC). Therefore, a) simultaneous training on separate deterministic tasks with the purpose of encoding motion primitives in a neural network, and b) the employment of maximally sparse rewards in combinations with virtual actuator constraints on velocity in setpoint proximity are advocated. For feedforward controller parametrization, both fully connected (FC) and recurrent neural networks (RNNs) are used.

2.2.1 Introduction

There exists a plethora of motion planning and control techniques for self-driving vehicles [300]. The diversity is caused by a core difficulty: the trade-off between model complexity and permitted online computation at short sampling times. In the following, three popular control classes for automated vehicles and recent vision-based end-to-end solutions are briefly summarized, before the proposed method is motivated.

Model-based Control Methods

In [208], a *sampling-based* anytime algorithm RRT* is discussed. Key notion is to *refine* an initial suboptimal path *while* it is followed. As demonstrated, this is feasible when driving towards a static goal in a static environment. However, it may be problematic in dynamic environments requiring to constantly replan paths, and where an online sampled suitable trajectory may not be returned in time. Other problems of *online* sampling-based methods are a limited model complexity and their tendency to produce jagged controls that require a smoothing step, e.g., via conjugate gradient [95].

In [273], a *lattice-based* method is discussed. Such methods, and similarly also based on *motion primitives* [118], [331], [163], [255], are always limited by the size of the look-up table that can be searched in real-time. In [273], a GPU is used for search.

In [106], *linear time-varying model predictive control* (LTV-MPC) is discussed for autonomous vehicles. While appealing for its ability to incorporate constraints, MPC must trade-off model-complexity vs. computational burden from solving optimization problems online. Furthermore, MPC is dependent on state and input reference trajectories, at least for linearization of dynamics, but almost always also for providing a tracking reference. Therefore, a two-layered approach is often applied, with motion planning and tracking as the two layers [300]. See Section 2.1.14 for a method using geometric corridor planning in the first layer for reference generation and for the combinatorial decision taking on which side to overtake obstacles. As indicated in [106, Sect. V-A] and further emphasized in Section 2.1.3, the selection of reference velocities can become problematic for *time-based* MPC and motivated to use *spatial-based* system modeling. Vehicle dynamics can be incorporated by inflating obstacles [163]. For tight maneuvering, the linearization approach from Section 2.1.10 is more accurate, however, computationally more expensive.

To summarize, two core observations are made. First, all methods (from sampling-based to MPC) are derived from vehicle *models*. Second, all of above methods suffer from the real-time requirement of short sam-

pling times. As a consequence, all methods make simplifications on the employed model. These include, for example, omitting of dynamical effects, tire dynamics, vehicle dimensions, using inflated obstacles, pruning search graphs, solving optimization problems iteratively, or offline precomputing trajectories.

Vision-based Methods

In [308], a pioneering end-to-end trained neural network labeled ALVINN was used for steering control of an autonomous vehicle. Video and range measurements are fed to a *fully connected* (FC)-network with a single hidden layer with 29 hidden units, and an output layer with 45 direction output units, i.e., *discretized* steering angles, plus one road intensity feedback unit. ALVINN does not control velocity and is trained using supervised learning based on road "snapshots". Similarly, recent DAVE-2 [63] also only controls steering and is trained supervisedly. However, it outputs *continuous* steering action and is composed of a network including *convolutional neural networks* (CNN) as well as FC-layers with a total of 250000 parameters. During testing (i.e., after training), steering commands are generated from *only* a front-facing camera. Another end-to-end system based on only camera vision is presented in [80]. First, a driving intention (change to left lane, change to right lane, stay in lane and break) is determined, before steering angle is output from a *recurrent neural network* (RNN). Instead of mapping images to steering control, in [79] and [374], *affordance indicators* (such as distance to cars in current and adjacent lanes etc.) and feasible driving actions (such as straight, stop, left-turn, right-turn) are output from neural networks, respectively. See also [306] and their treatment of "option policies". To summarize, it is distinguished between a) vision-based *end-to-end* control, and b) *perception-driven* approaches that attempt to extract useful features from images. Note that such features (e.g., obstacle positions) are implicitly required for all of the aforementioned model-based control methods.

Motivation and Contribution

As noted in [360], correct localization relative to lane boundaries is more important than with respect to GPS-coordinates. This indicates the importance of lasers, lidars and cameras for automated driving. Second, vehicles are man- and woman-made products for which there exist decade-long experience in vehicle dynamics modeling [131],[317]. There is no sensible reason to a priori entirely discard this knowledge (for manufacturers it is present even in form of construction plans). Thus, available vehicle models should be leveraged for control design. Consider also the position paper [132] for general limitations of end-to-end learning. Third, a general purpose control setup is sought avoiding to switch between different vehicle models and algorithms for, e.g., highway driving and parking. There also exists only one real-world vehicle. In that perspective, the most complex vehicle model encompassing all driving scenarios is in general preferable for control design. Also, a model mismatch on the planning and tracking layer can incur paths infeasible to track [163]. Fourth, the most accident causes involving other mobile vehicles are *rear-end* collisions [289], which most frequently are caused by inattentiveness or too close following distances. In [155] it was found that reaction delays of a single sampling time can already decide upon crash avoidance or failure. Therefore, a control method that enables minimal sampling times, such as feedforward control, can significantly increase safety. Importantly, this is achieved deterministically. In contrast, environment motion prediction (which also can increase safety) always remains stochastic. Fifth, small sampling times are contradicting the preference for complex vehicle models. These considerations motivate to use *neural network controllers*, which enable:

1. *Offline* training on arbitrarily sophisticated high-fidelity models in arbitrarily complex simulation environments.

2. *Online* fast network feedforward evaluation.

Because of these two characteristics, neural network controllers have the potential to eventually also be employed in nano-scale robots [234]. In

an automated vehicles settings, it implies that once trained, low-cost embedded hardware can be used online with only a few matrix vector multiplications being evaluated forward.

2.2.2 Problem Formulation

Let vector $z_t = [x_t,\ y_t,\ \psi_t,\ v_t]$ summarize four of the vehicle's states. Coordinates x_t and y_t describe the center of gravity (CoG) in the inertial frame, ψ_t denotes the yaw angle relative to the inertial frame, and v_t indicates the vehicle's speed at time t. Let $z^{\text{goal}} = [x^{\text{goal}},\ y^{\text{goal}},\ \psi^{\text{goal}},\ v^{\text{goal}}]$ denote a goal state. Then, the following problem can be formulated.

Problem 1. *Given vector z_t at time t, determine a general purpose control method that enables closed-loop navigation from z_t to z^{goal}, accounting for physical actuator constraints, and ensuring obstacle avoidance.*

Thus, a general purpose control method that can serve the entire range of driving operations is sought. These include road following (RF), intersection control (IC), static and dynamic obstacle avoidance (OA), adaptive cruise control (ACC) and parking applications. Four main classes are distinguished:

- Zone navigation (no obstacles, no road bounds);
- Road navigation (static obstacles, road bounds);
- Adaptive cruise control (ACC) on top of road navigation;
- Dynamic obstacle avoidance on top of road navigation.

The choice of four-state z_t is motivated by this distinction, since instances of all four classes can be characterized by an associated z^{goal} and a deviation measure relating z_t to z^{goal}. In addition, z^{goal} can be reassigned recursively such that it is rendered *time-varying* z_t^{goal}. This enables to encode IC as well as ACC applications.

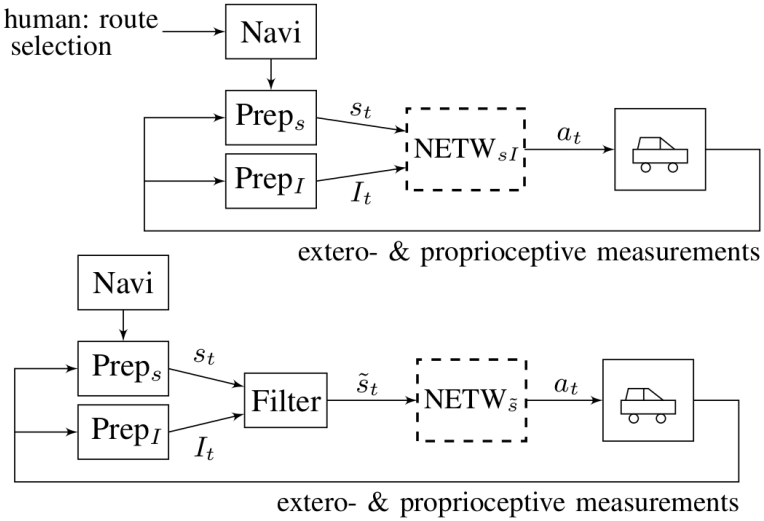


Figure 24: Two candidates for the closed-loop control system architecture.

2.2.3 System Level

Closed-Loop Control System

Two candidate architectures are considered, see Figure 24. There are four core components common to both. The first is a neural network acting as the controller, abbreviated as "NETW" and subscripted by $_{\tilde{s}}$ and $_{sI}$, respectively. They differ by their input vector. Thus,

$$s_t = \left\{ z_\tau - z_\tau^{\text{goal}} \right\}_{\tau=t-\Gamma}^{t}, \tag{2.50}$$

$$I_t = \{ J_\tau \}_{\tau=t-\Gamma}^{t}, \tag{2.51}$$

are defined, where hyperparameter $\Gamma \geq 0$ denotes the number of past states considered, and where J_τ is a pixel-related multi-dimensional array. In the general case, z_τ^{goal} is time-varying. However, for e.g. parking applications, it is static $z_\tau^{\text{goal}} = z^{\text{goal}}, \forall \tau$. "Filter" denotes an algebraic mapping from (s_t, I_t) to $\tilde{s}_t$. For example, in the simplest case, $\tilde{s}_t = s_t$. The block output of NETW is a control signal a_t. The entire

Section 2.2.4 is devoted to the design and training of NETW. The second component in Figure 24 is "Navi" (navigation planner). It outputs a high-level route plan. The third and fourth block component are labeled as "Prep$_s$' and "Prep$_I$" since they receive extero- and proprioceptive sensor measurements, process these, and prepare them (hence abbreviation "Prep") such that s_t and I_t accordig to (2.50) and (2.51) can be fed to the control network. Exteroceptive measurements are assumed to include inter-vehicular communication (car-2-car) sensings as well as the communication with a centralized or decentralized coordination service such that, in general, multi-automated vehicle coordination is also enabled [154].

Location, Heading and Velocity-related System Component

The choice of s_t in (2.50) is discussed. Nonlinear dynamic vehicle models that include tire dynamics can be very complex with up to a double-digit number of states [317]. For all kinematic and dynamic vehicle models, and of varying complexity, the goal vehicle pose, within a time-varying and a static setting, can be described well by only location, heading and velocity, i.e., $z_\tau^{\text{goal}} = [x_\tau^{\text{goal}},\ y_\tau^{\text{goal}},\ \psi_\tau^{\text{goal}},\ v_\tau^{\text{goal}}]$. This entails the question of what to provide as input to the network controller: a high-dimensional state according to the vehicle model used for training, or only a four-dimensional $z_\tau = [x_\tau,\ y_\tau,\ \psi_\tau,\ v_\tau]$. It is opted for the latter option. This is motivated as follows. Because of receiving reward measures as a function of z_τ^{goal} during training, the network controller naturally encodes vehicle dynamics when mapping s_t and I_t (or $\tilde{s}_t$) to a_t. It is not obvious how providing a more detailed state (e.g., including various *rate* states) as input to the network would improve this encoding since the goal pose can already be described accurately by the only four-dimensional z_τ^{goal}.

Pixel-related System Component

This section discusses I_t in (2.51). It is proposed to map all exteroceptive measurements to *one sparse multi-channel* image. One of these channels describes all area prohibited from trespassing (obstacles) within a *range-*

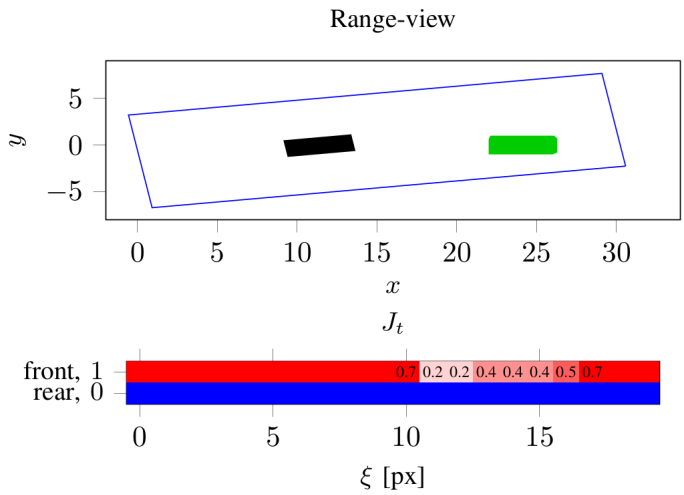


Figure 25: (Top) Illustration of the range-view concept. Black indicates the ego vehicle and green a detected obstacle. (Bottom) Illustration of $J_t \in \mathbb{R}^{2\times N_\xi}$ with matrix entries normalized along the length (longer rectangle edge) of the range-view. Colors are displayed for emphasis.

view in the proximity of the vehicle, see Figure 25. This permits general purpose driving according to the four classes described in Section 2.2.2. This approach also exploits the fact that vehicles operate ground-based. Thus, for all obstacle avoidance tasks the area prohibited from trespassing can be described accurately in the 2D plane. Image-sparsity naturally results from the distinction between area permitted (pixel-value zero) and area prohibited to drive in (pixel value greater than zero). It also permits to distinguish between obstacles of different kind (static, dynamic, bicycles, vehicles etc.) by assigning different pixel values. Third, a sparse multi-channel image is a well-suited input for training of *location-aware* neural networks (such as fully connected ones). Fourth, it guarantees a fixed-size complexity of I_t (fixed pixel resolution) fed to the neural network controller, while at the same time permitting an adaptable, e.g., velocity-dependent range-view size. Thus, complexity is always bounded by the number of pixels, $N_\eta \times N_\xi$, employed to represent the range-view.

In addition, a simplification of the sparse multi-channel image is considered. The first layer of the I_t-related part of the network controller (NETW) receives an input vector of dimension $N_I = (1+\Gamma)N_c N_\xi N_\eta$. Dependent on the image resolution and since typically $N_\eta > N_\xi$, the number N_I can be quite large. Therefore, an alternative I_t is proposed, where each image-frame, $\tilde{J}_{\tau,c} \in \mathbb{R}^{N_\eta \times N_\xi}$, is replaced by a reduced $J_{\tau,c} \in \mathbb{R}^{2\times N_\xi}$ array. Thus, it is defined

$$J_{\tau,c,1,\xi} = \arg\min_{\eta} \left\{ \tilde{J}_{\tau,c,\eta,\xi} > 0 \right\}_{\eta=N_\xi^{\text{CoG}}}^{N_\eta} - K_\xi^{\max}, \tag{2.52a}$$

$$J_{\tau,c,0,\xi} = -K_\xi^{\min} + \arg\max_{\eta} \left\{ \tilde{J}_{\tau,c,\eta,\xi} > 0 \right\}_{\eta=0}^{N_\xi^{\text{CoG}}}, \tag{2.52b}$$

with $K_\xi^{\max} = \max(N_\xi^{\text{CoG}}, N_\xi^{\text{veh,max}})$ and $K_\xi^{\min} = \min(N_\xi^{\text{CoG}}, N_\xi^{\text{veh,min}})$ for all $\tau = t-\Gamma, \ldots, t$, $c = 1, \ldots, N_c$, $\xi = 0, \ldots N_\xi - 1$. Thus, the signed minimum pixel distances between vehicle chassis, $N_\xi^{\text{veh,max}}$ and $N_\xi^{\text{veh,min}}$, or CoG, N_ξ^{CoG}, and closest obstacle in forward and backward facing directions are represented, see Figure 25. The multi-channel description is maintained, such that, one channel can describe distance, while another the obstacle-type (static, dynamic, etc.) or road surface. It is further argued that network learning capability is *not* affected when enabling *temporal* data processing (i.e., with memory). This can be seen when envisioning how the network controller can learn to pass by obstacle corners. Temporal perception processing enables to learn anticipative and, in particular, outreaching steering. This motivates that either a) *multilayer perceptrons* (MLPs) with $\Gamma > 0$ are employed, or, b) and alternatively, recurrent neural networks (RNNs).

The sensor fusion of all exteroceptive measurements to obtain a unified I_t-array is the second main task besides design of a learning algorithm to train the network. It includes a transformation step from various physical measurement units to pixel-coordinates. See [258] for *semantic segmentation* based on *fully convolutional networks* (FCN). Importantly, note that a) sensor fusion to obtain I_t, and b) the training of the network controller can be solved *orthogonally*. The former problem is of *supervised* machine learning nature. In contrast, the latter represents a *reinforcement*

learning problem to which the entire Section 2.2.4 is devoted to.

Control Vector

The *continuous* control vector is here defined as

$$a_t = [v_t,\ \delta_t]. \tag{2.53}$$

Both v_t and δ_t are subject to physical actuator absolute and rate constraints. These are treated as *part of the vehicle model* on which the network training is based on (see the next Section). Note that the minimum velocity is negative. Hence, reverse driving is explicitly permitted. The network output, which results from a $\tanh(\cdot)$ activation function as will be further specified below, is linearly scaled to be within *absolute* actuator bounds, before it is fed to the low-level controllers that are further limited by actuator *rate* constraints.

A remark with respect to gear selection can be made. Throughout this section, electric vehicles (EVs) are assumed since they appear more suitable to curb urban pollution. EVs do not require gearboxes. Nevertheless, the network architecture can be extended to include discrete gear as an additional decision variable. Suppose N_{gears} gears are available. Then, the output layer can be extended by N_{gears} channels, with each channel output representing a normalized probability of gear selection as a function of s_t and I_t (or $\tilde{s}_t$), that can be trained by means of a *softmax classifier*.

In fact, the last discussion further underlines the potential of model-based reinforcement learning using neural networks. Similarly, *mixed-integer* decisions such as on which side to overtake an obstacle at what speed can in principle be handled efficiently using the neural approach.

Exemplary System Model used for Learning

A fundamental motivation of proposed method is to leverage decade-long experience in vehicle modeling [131], [317]. Since vehicles are man- and woman-made products, there is no sensible reason to entirely a priori discard existing vehicle model equations and construction insights.

For reduced computational burden in this preliminary proof of concept, a simple nonlinear bicycle model is employed for the experiments in Section 4.4.5. Note, however, that in practice it is recommended to employ a high-fidelity vehicle model. Physical actuator absolute and rate constraints are treated as part of the vehicle model on which the network training is based on. The deterministic system model assumes

$$v_t \leq \min(v_t,\ v_{t-1} + \dot{v}_{\max,t} T_s,\ \tilde{I}^{\max}_{v,t-1} v_{\max,t}) \tag{2.54a}$$

$$v_t \geq \max(v_t,\ v_{t-1} + \dot{v}_{\min,t} T_s,\ \tilde{I}^{\min}_{v,t-1} v_{\max,t}) \tag{2.54b}$$

with $\tilde{I}^{\max}_{v,t-1} = 1$ if $v_{t-1} \geq 0$ and 0 otherwise, and $\tilde{I}^{\min}_{v,t-1} = 1$ if $v_{t-1} \leq 0$ and 0 otherwise. The two indicators are introduced to guarantee a zero-crossing w.r.t. v-control to remain for at least one T_s at zero velocity. Furthermore, there is $\delta_t \leq \min(\delta_t, \delta_{t-1} + \dot{v}_{\max,t} T_s, \delta_{\max,t})$, $\delta_t \geq \max(\delta_t, \delta_{t-1} + \dot{\delta}_{\min,t} T_s, \delta_{\max,t})$, and a simple Euler-discretized nonlinear kinematic bicycle model with $x_{t+1} = x_t + T_s v_t \cos(\psi_t)$, $y_{t+1} = y_t + T_s v_t \sin(\psi_t)$, $\psi_{t+1} = \psi_t + T_s \frac{v_t}{b} \tan(\delta_t)$, $v_{t+1} = v_t$, modeled with $b = 3.5$m, and rectangular vehicle dimenstions of 1.8×4.3m.

General Purpose Method

At mission initialization, a navigation route is computed. Then, the following basic algorithm can be conducted:

1. Determine (z_t, I_t) from a mapping of proprio- and exteroceptive sensors to account for road boundaries, static and dynamic obstacles, traffic lights and signs.
2. Determine $f_t^{\text{ACC}} \in \{0, 1\}$, a flag for ACC-possibility.
3. If $f_t^{\text{ACC}} == 1$, then apply conservative driving with

$$z_t^{\text{goal}} = \begin{cases} z_t^{\text{stop}}, & \text{if } \exists \text{ red traffic light/sign ahead}, \\ z_t^{\text{lead}}, & \text{if } d_t < d^{\min}(v_t, v_t^{\text{lead}}), \\ z_t^{\text{horizon}}, & \text{otherwise}, \end{cases}$$

   otherwise apply emergency obstacle avoidance with $z_t^{\text{goal}} = z_t^{\text{distant}}$ and *relaxed* permissible driving area.

4. Evaluate NETW as a function of s_t and I_t to obtain a_t.

5. Apply a_t and wait until the next sampling time.

Thus, three parameterized *decisions* need to be made. First, on the possibility of ACC. This decision can typically be parameterized as a function of current and leading vehicle state. The second decision is about a still tolerable minimum safety distance $d_t < d^{\min}(v_t, v_t^{\text{lead}})$ with respect to a leading vehicle characterized by state z_t^{lead}. See [155] for a simple heuristical parameterization of both of the first two decisions. The third decision involves a selection of a suitable goal state in case of an emergency obstacle avoidance scenario. Therefore, a vehicle state z_t^{distant} located *distant* away but along the reference path may heuristically be imposed. This permits the control system to fully concentrate on immediate obstacle avoidance rather than tracking of a reference state. Relaxation of permissible driving area implies to also consider neighboring lanes for a potential obstacle avoidance maneuver. In contrast to above heuristic decisions, z_t^{stop} with $v_t^{\text{stop}} = 0$ is prescribed by traffic lights and stop signals. For road tracking, the goal state is set as z_t^{horizon}, i.e., the intersection of the navigation route and the range-view.

Discussion and Comparison with Model Predictive Control

Before presenting the learning algorithm, main differences to MPC are stressed. First, implicit MPC solves an optimization problem *online*. This limits the complexity of the predictive models that can be used. In contrast, neural network controllers can be trained offline on the most sophisticated vehicle models, before being employed online in a computationally cheap feedforward evaluation. As a consequence, online solvers for MPC are not required anymore.

Second, it is also mentioned *explicit* MPC [41], which reduces online burden to search in a look-up table as a funcion of current state (MPC problems are parameterized as a function of state and solved offline). Explicit MPC is limited to a) *time-invariant* models (and thus unsuited for general navigation tasks), b) still requires an online search of the look-up table storing the state-dependent controls, and c) only suitable for small

problems. For perspective, in [368] for problems with more than approximately 5 states, 3 controls and 12 inequality constraints, explicit MPC is considered to no longer be practically feasible. It may be interesting to analyze that simple explicit (linear) MPC applications can be replaced by neural network feedforward controllers, which are a) trained on more complex (nonlinear) system models using reinforcement learning and b) are still faster to evaluate online.

Third, spatial-based MPC can be superior to time-based MPC due to its ability to easily constrain vehicle motion to a convex driving corridor expressed along a spatially discretized road centerline. In proposed neural network setting, this driving corridor is represented by a pixel image. Importantly and in contrast to MPC, convexity of permissible driving area is not a prerequisite anymore. This, in general, liberates from the convex corridor planning step preceding MPC, see [155] and the references therein.

Fourth, while MPC optimizes over a *predictive* model, the nature of neural network controllers appears of more *reactive* nature, eventhough predictive capability is encoded in the network through training and receding z^{goal}-selection during online operation. The most frequent accident cause involving other mobile vehicles are *rear-end* collisions [289], which most frequently result from inattentiveness or too close following distances. In this perspective, another interesting topic is to argue to what extend *fast reaction* times are more important in automated vehicles than predictive capability.

Finally, in view of current (and further developing) hardware opportunities, expensive offline training clearly is appropriate. To stress remarkable dimensions, in [327] training is distributed on 80 machines and 1440 CPU cores. Even more remarkable, in [4] 1024 Tesla P100 GPUs are used in parallel. For perspective, one Tesla P100 permits a double-precision performance of 4.7 TeraFLOPs [296].

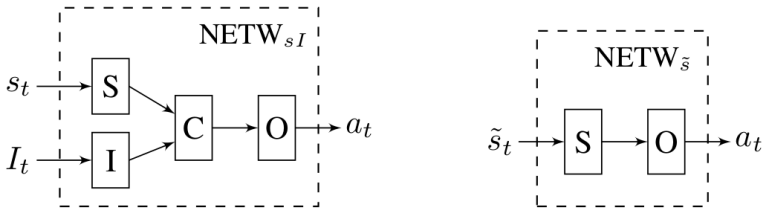


Figure 26: The two network structures of the control block candidates in Figure 24. It is distinguished between a S ($s_t/\tilde{s}_t$-related), I (I_t-related), C (combined) and O (output layer) part of the neural network. Each of the four parts can be composed of an arbitrary sequence of FCs and RNNs.

2.2.4 Training Algorithm

This section proposes a method to train neural network controllers.

Network Controller

Two network structure candidates are displayed in Figure 26. Alternative structures were also tested, but did not improve performance, or only increased the total number of weights to be learnt (e.g., when decoupling controls and not sharing weights). Each of the four parts displayed in Figure 26 can be composed of an arbitrary sequence of FCs, LSTM cells including peephole connections [129],[162], and GRUs [81]. Motivation for these choices are the simplicity of FCs and the temporal processing ability of RNNs. All parameter weights to be learnt are initialized by Gaussian-distributed variables with zero mean and a small variance σ^2_{init}. Exceptions are adding a 1 to the LSTMs forget gate biases for LSTM cells, as recommended in [201], which are thus initialized with mean 1. In proposed setting, the affine part of all FC-layers is followed by nonlinear $\tanh(\cdot)$ activation functions acting elementwise. Because of their bounded outputs, saturating nonlinearities are preferred over ReLUs, which are used for the hidden layers in other RL settings [253] but can result in large unbounded layer output changes. Before entering the first layers of the S- and I-block, s_t and I_t are normalized elementwise, whereby for final experiments 30m, 2π, $v_{\text{max}} - v_{\text{min}}$ and N_η are heuris-

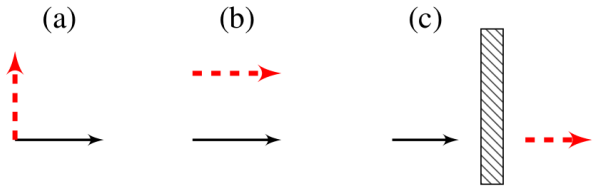


Figure 27: The problematic of rich rewards. Three scenarios (a), (b) and (c) indicating different start (black) and goal (red dashed) states (position and heading). For (c), an obstacle is added.

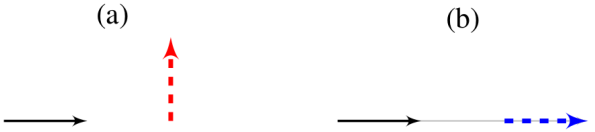


Figure 28: Curriculum Learning. The difficulty of selecting "simple" examples. The original problem with start (black) and goal (red dashed) state is denoted in (a). A "simpler" problem is given in (b). See Section 2.2.4 for interpretation.

tically selected as normalization constants for distances x_t and y_t, angle ψ_t, velocity v_t and all elements of I_t, respectively. Their choice was not found to be decisive for learning. The final FC-layer comprises a $\tanh(\cdot)$ activation function. It accordingly outputs two bounded continous values referred to as a_t^{NN}, which are then affinely scaled to a_t as outlined in the next section. Then, during training a_t is fed to the system model.

Reward Shaping

Reward shaping is crucial for the success of learning by reinforcement signals [349]. However, it was found to be a highly delicate matter in practical problems. Therefore, the preferred choice is motivated in detail.

In most practical control problems, a current state z_0 is given at $t = 0$, and a desired goal state z^{goal} is known. Not known, however, is the shape of the best trajectory (w.r.t. a given criterion) and the control signals that realize that trajectory. Thus, by nature these problems offer a *sparse* reward signal, $r_{\tilde{T}}(z^{\mathrm{goal}})$, received only upon reaching the desired goal state

at some time $\tilde{T} > 0$. In the following, auxiliary *rich reward* signals and *curriculum learning* [44] are first discussed.

A reward signal $r_t(z_t, a_t)$, abbreviated by r_t, is labeled as *rich* when it is *time-varying* as a function of states or controls. Note that the design of any such signal is heuristic and motivated by the hope for accelerated learning through maximally frequent feedback. In the following, the problematic of rich rewards is exposed. First, $e_{\psi,t} = |\psi_t - \psi^{\text{goal}}|$, $e_{d,t} = \sqrt{(x_t - x^{\text{goal}})^2 + (y_t - y^{\text{goal}})^2}$, and $e_{v,t} = |v_t - v^{\text{goal}}|$ are defined, as well as the binary flag indicating whether the goal state is reached,

$$f_t^{\text{goal}} = \begin{cases} 1, & \text{if } (e_{d,t} < \epsilon_d) \wedge (e_{\psi,t} < \epsilon_\psi) \wedge (e_{v,t} < \epsilon_v), \\ 0, & \text{otherwise,} \end{cases} \tag{2.55}$$

where $(\epsilon_d, \epsilon_\psi, \epsilon_v)$ are small tolerance hyperparameters. Then, suppose a rich reward signal of the form $r_t = -(\alpha e_{d,t} + e_{\psi,t})$ is designed, which characterizes a weighted linear combination of different measures. This class of reward signals, trading-off various terms and providing feedback at every step, occurs frequently in the literature [318], [228], [253], [178]. However, as will be shown, in an automotive setting, it may easily lead to undesirable behavior. Suppose case (a) in Figure 27 and a maximum simulation time $T^{\max}$. Then, omitting a discount factor for brevity, $-T^{\max}\frac{\pi}{2} > -\sum_{t=0}^{\tilde{T}} \alpha e_{d,t}^\star + e_{\psi,t}^\star$, may be obtained for accumulated rewards. Thus, the no-movement solution may incur more accumulated reward, namely $-T^{\max}\frac{\pi}{2}$, in comparison to the true solution, which is indicated on the right-hand side of the inequality sign.

Similarly, for specific $(\alpha, T^{\max})$, the second scenario (b) in Figure 27 can return a no-movement solution since the initial angle is already coinciding with the target angle. Hence, for a specific $(\alpha, T^{\max})$-combination, the accumulated reward when not moving may exceed the value of the actual solution.

The third scenario (c) in Figure 27 further illustrates that even if reducting rich rewards to a single measure, e.g., $r_t = -e_{d,t}$, an undesired standstill may result. This occurs especially in the presence of obstacles (and maze-like situations in general).

To summarize, for finite $T^{\max}$, the design of auxiliary *rich* reward sig-

nals is not straightforward. It can easily result in solution trajectories that may even be *globally optimal* w.r.t. accumulated reward, however, prohibit to solve the original problem of determining a trajectory from initial to target state.

In [44], curriculum learning (CL) is discussed as a method to speed up learning by providing the learning agent first with *simpler* examples before gradually increasing complexity. Analogies to humans and animals are drawn. The same paper also acknowledges the difficulty of determining "interesting" examples [44, Sect. 7] that optimize learning progress. Indeed, CL entails the following issues. First, "simpler" tasks need to be identified. This is not straightforward as discussed shortly. Second, these tasks must first be solved before their result can serve as initialization to more complex tasks. In contrast, without CL, the entire solution time can be devoted to the complex tasks rather than being partitioned into easier and difficult tasks. In experiments, this was found to be relevant. Third, the solution of an easier task does not necessarily represent a better initialization to a harder problem in comparison to an alternative random initialization. For example, consider the scenario in Figure 28. The solution of the simpler task does not serve as a better initialization than a purely random initialization of weights. This is since the final solution requires outreaching steering and possibly reversing of the vehicle. The simpler task just requires forward driving and stopping. This simple example illustrates the need for careful manual selection of suitable easier tasks for CL.

In the course of this work, a plethora of reward shaping methods were tested. These include, first, solving "simpler" tasks by first dismissing obstacles and target angles limited to $30°$-deviation from the initial heading. Second, ϵ-tolerances were initially relaxed before gradually decreasing them. Third, it was tested to first solve a task for only the ϵ_d-criterion, then both $(\epsilon_d, \epsilon_\psi)$, and only finally all of $(\epsilon_d, \epsilon_\psi, \epsilon_v)$. Also varying sequences (e.g., first ϵ_ψ instead of ϵ_d) were tested. No consistent improvement could be observed for neither of these methods. On the contrary, solving allegedly simpler tasks reduced available solve time for the original "hard" problems. Without CL the entire solution time can be

devoted to the complex tasks. Therefore, the preferred reward design is:

$$r_t = \begin{cases} -\infty, & \text{if } f_t^{\text{crash}} == 1, \\ -1, & \text{otherwise}, \end{cases} \tag{2.56}$$

in combination with f_t^{goal} from (2.55), and

$$\Delta p_t = -\sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2}, \tag{2.57}$$

$$F_i = \begin{cases} 1, & \text{if } \sum_{\tau=t-T^{\text{goal}}+1}^{t} f_\tau^{\text{goal}} == T^{\text{goal}}, \\ 0, & \text{otherwise}, \end{cases} \tag{2.58}$$

where f_t^{crash} is an indicator flag of a vehicle crash. Upon $F_i = 1$, the reinforcemet learning problem is considered as solved. For generality, an integral F_i is defined with problems such as the *inverted pendulum* [10] in mind, which require stabilization and are only considered to be solved after stabilization is demonstrated for sufficiently many consecutive T^{goal} time steps. Note, however, that this is not required for an automotive setting. Here, it must be $T^{\text{goal}} = 1$. Only then learning with $v^{\text{goal}} \neq 0$ is possible (see also the discussion of the main algorithm below). Other criteria and trade-offs for Δp_t naturally are possible, especially, accumulated curvature of resulting paths, and a minmax objective therefore. The negation is introduced for maximization ("hill climbing" convention). Note that the preferred reward signal is maximal sparse, returning a negative constant, -1, for all times up until reaching the target. It represents a *tabula rasa* solution critizised in [318] for its maximal sparsity. Indeed, standalone it was not sufficient to facilitate learning when also accounting for a velocity target v^{goal}. Therefore, *virtual constraints* (VC) on velocity in target proximity are introduced. Thus,

$$v_t^m = \begin{cases} v^m, & \text{if } e_{d,t} \geq R_v^{\text{thresh}}, \\ v^{\text{goal}} + \frac{(v^m - v^{\text{goal}})}{R_v^{\text{thresh}}} e_{d,t}, & \text{otherwise}, \end{cases} \tag{2.59}$$

is defined, where $m \in \{\max, \min\}$, and R_v^{thresh} is a hyperparameter (e.g., the range-view length or a heuristic constant). Then, the neural network output is scaled as

$$a_t = \begin{bmatrix} \frac{v_t^{\max} - v_t^{\min}}{2} \\ \frac{\delta^{\max} - \delta^{\min}}{2} \end{bmatrix} \odot a_t^{\text{NN}} + \begin{bmatrix} \frac{v_t^{\max} + v_t^{\min}}{2} \\ \frac{\delta^{\max} + \delta^{\min}}{2} \end{bmatrix} \tag{2.60}$$

where $\odot$ here denotes the Hadamard product. Let us legitimize VCs. Since speed is a decision variable it can always be artificially constrained. This justifies the introduction of (2.59). Here, bounds are set to *affinely* converge towards v^{goal} in the proximity of the goal location. This is a heuristic choice. Note that the affine choice does *not* necessarily imply constant accelerations. This is since (2.59) is spatially parameterized. Note further that rate-constraints (2.54) still hold when (2.60) is applied to the vehicle. For $R_v^{\text{thresh}} = 0$, VCs are dismissed. It was also tested to constrain δ_t. However, this did not accelerate learning. The final heading pose implies circles prohited from trespassing because of the nonholonomic vehicle. It was also tested to add these as *virtual obstacles*. However, this too did not improve. Finally, note that VCs on velocity artificially introduce hard constraints, parameterized by R_v^{thres}, and thus shape the learning result w.r.t velocity, at least *towards the end* of the trajectory. Two comments are made. First, in receding online operation, with additional frequent resetting of targets, this shaping effect is reduced since only the first control of a planned trajectory is applied. Second, the influence of hyperparameter R_v^{thres} only becomes apparent during parking when following the trajectory up until standstill. However, here no significant velocity changes are desired, such that the R_v^{thres}-choice is not decisive.

To summarize this section. It was illustrated that the design of *rich reward* signals as well as *curriculum learning* can be problematic. Therefore, *maximal sparse rewards* in combination with *virtual constraints* on velocity were proposed.

The Role of Tolerances

Tolerances ϵ hold an important role. On one hand, nonzero ϵ-tolerances result in deviations between actually learnt $\hat{z}^{\text{goal}}$ and originally desired goal pose z^{goal}. On the other hand, very small ϵ (e.g., $\epsilon_d = 0.1\text{m}$, $\epsilon_\psi = 1^\circ$ and $\epsilon_v = 1\text{km/h}$) prolong learning time. Two scenarios apply.

First, for a network trained on a large-scale and dense grid of training tasks and for small ϵ, during online operation, suitable control commands are naturally interpolated even for setpoints not seen during train-

ing. This applies especially when training and online operation are conducted *obstacle-free*. Here, structured task design can be applied optimally, gridding tasks over polar coordinates and velocity. The concept of interpolation through *encoded motion primitives* within the network is the core advantage over methods relying on look-up tables with stored trajectories, and which require to solve time-critical search problems. For example, in [273] exhaustive search of the entire lattice-graph is conducted online on a GPU. In [255], a total of about 100 motion primitives is considered. Then, online an integer program is solved by enumeration using maximal progress along the centerline as criterion for selection of the best motion primitive. In contrast, for neural feedforward control this search is not required.

Second, the scenario is considered in which existing training hardware does a) not permit large-scale encoding, and b) only permits to use larger ϵ-tolerances to limit training time. Therefore, the following method is devised. First, tuples $(\hat{z}^{\text{goal}}, z^{\text{goal}})$ are stored for each training task. Then, during online operation, for any setpoint, z^{setpoint}, the closest (according to a criterion) $\hat{z}^{\text{goal}}$ from the set of training tasks is searched, before the corresponding z^{goal} is applied to the network controller. Two comments are made. First, in order to reach $\hat{z}^{\text{goal}}$ (with zero deviation), z^{goal} must be applied to the network. Therefore, *tuples* need to be stored. Second, eventhough this method now also includes a search, it still holds an important advantage over lattice-based methods. Namely, through the *compression* of the look-up table in the network weights. Hence, only tuples need to be stored—not entire trajectories. This is especially relevant in view of limited *hardware memory*. Thus, through encoding, potetially many more motion primitives can be stored in comparison to lattice-based methods.

In practice, the first scenario is obviously preferrable. It is also implementable for two reasons. First, recall aforementioned computational opportunities. Second, neural networks have in principle unlimited function approximation capability [340]. Hence, the implementation of the first approach is purely a question of intelligent task setup, and computational power.

Algorithm 4: Task Separation with Hill Climbing (TSHC)

Input: system model, network structure, training tasks, and N_{restarts}, $N_{\text{iter}}^{\max}$, n, N_{tasks}, $T^{\max}$, β, ϵ, $\sigma_{\text{pert}}^{\min}$, $\sigma_{\text{pert}}^{\max}$.
Initialize $\theta^\star \leftarrow \emptyset$, $N^\star \leftarrow 0$, $P^\star \leftarrow -\infty$, $J^\star \leftarrow 0$.
for $i_{restart} = 1, \ldots, N_{restarts}$ **do**
  Initialize θ randomly, and $\sigma_{\text{pert}} \leftarrow \sigma_{\text{pert}}^{\max}$, $N_{\text{old}}^{\text{tasks},\star} \leftarrow 0$.
  for $i_{iter} = 1, \ldots, N_{iter}^{max}$ **do**
    % RUN ASYNCHRONOUSLY:
    for $i = 1, \ldots, n$ **do**
      Perturb $\theta_i \leftarrow \theta + \sigma_{\text{pert}}\zeta$, with $\zeta \sim \mathcal{N}(0, I)$.
      Initialize $N_i^{\text{tasks},\star} \leftarrow 0$, $P_i \leftarrow 0$, $J_i \leftarrow 0$.
      for $i_{task} = 1, \ldots, N_{tasks}$ **do**
        Initialize z_0 (and LSTM and GRU cells).
        for $t = 0, \ldots, T^{max} - 1$ **do**
          Read (s_t, I_t) from i_{task}-environment, $a_t \leftarrow \mathcal{X}(s_t, I_t, \theta_i)$.
          $(r_t, \Delta p_t, f_t^{\text{goal}}) \leftarrow \mathcal{R}(s_t, I_t, a_t)$, $z_{t+1} \leftarrow \mathcal{Z}(z_t, a_t)$.
          $P_i \leftarrow P_i + \Delta p_t$, $J_i \leftarrow J_i + r_t$, F_i according to (2.58).
          if $(F_i == 1) \vee (r_t == -\infty)$ **then**
            Break t-loop.
        $N_i^{\text{tasks},\star} \leftarrow N_i^{\text{tasks},\star} + F_i$.
    % DETERMINE $i^\star$:
    if $\max_i \{N_i^{tasks,\star}\}_{i=1}^n == N_{tasks}$ **then**
      $i^\star = \arg\max_i \left\{ P_i \mid N_i^{\text{tasks},\star} == N_{\text{tasks}} \right\}_{i=1}^n$.
      if $P_{i^\star} > P^\star$ **then**
        $(\theta^\star, N^\star, P^\star, J^\star) \leftarrow (\theta_{i^\star}, N_{\text{tasks}}, P_{i^\star}, J_{i^\star})$.
    else
      $i^\star = \arg\max_i \{J_i\}_{i=1}^n$.
      if $(J_{i^\star} > J^\star) \wedge (P^\star == -\infty)$ **then**
        $(\theta^\star, N^\star, P^\star, J^\star) \leftarrow (\theta_{i^\star}, N_{i^\star}^{\text{tasks}}, P_{i^\star}, J_{i^\star})$.
    $N^{\text{tasks},\star} \leftarrow N_{i^\star}^{\text{tasks},\star}$.
    % UPDATE PARAMETERS:
    if $N^{\text{tasks},\star} > N_{\text{old}}^{\text{tasks},\star}$ **then** $\sigma_{\text{pert}} \leftarrow \max(\frac{1}{\beta}\sigma_{\text{pert}},\ \sigma_{\text{pert}}^{\min})$.
    else if $N^{\text{tasks},\star} < N_{\text{old}}^{\text{tasks},\star}$ **then** $\sigma_{\text{pert}} \leftarrow \min(\beta\sigma_{\text{pert}},\ \sigma_{\text{pert}}^{\max})$.
    $\theta \leftarrow \theta_{i^\star}$ and $N_{\text{old}}^{\text{tasks},\star} \leftarrow N^{\text{tasks},\star}$.
    % OPTIONAL:
    if $N_i^{\text{tasks},\star} == N_{\text{tasks}}$ **then** Break i_{iter}-loop. % no further refinement.
Output: $(\theta^\star, N^\star, P^\star, J^\star)$.

Main Algorithm – TSHC

Algorithm 4 is proposed for simple gradient-free model-based reinforcement learning. The name is derived from the fact of a) learning from separate training tasks, and b) a hill climbing update of parameters (greedy local search).

Let us elaborate on definitions. Analysis is provided further below. First, all network parameters are lumped into variable θ. Second, the perturbation Step 8 in Algorithm 4 has to be intepreted accordingly. It implies parameter-wise affine perturbations with zero-mean Gaussian noise and spherical variance σ^2_{pert}. Third, $\mathcal{X}(\cdot)$, $\mathcal{R}(\cdot)$ and $\mathcal{Z}(\cdot)$ in Steps 13-14 denote functional mappings between properties defined in the preceding sections. Fourth, hyperparameters are stated in Step 1. While N_{restarts}, $N^{\text{max}}_{\text{iter}}$, n, N_{tasks} and T^{max} denote lengths of different iterations, $\beta > 1$ is used for updating of σ_{pert} in Step 30 and 31. Fifth, for every restart iteration, i_{restart}, multiple parameter iterations are conducted, at most $N^{\text{max}}_{\text{iter}}$ many. Sixth, in Steps 21 and 25 hill climbing is conducted, when a) all tasks have been solved for current i_{iter}, or b) not all tasks have yet been solved, respectively. Seventh, there are two steps in which an early termination of iteration may occur: Step 17 and 34. The former is a must. Only then learning with $v^{\text{goal}} \neq 0$ is possible. The latter termination criterion in Step 34 is optional. If dismissed, a *refinement step* is implied. Thus, eventhough all N_{tasks} tasks have been solved, parameter iterations (up until $N^{\text{max}}_{\text{iter}}$) are continued. Eight, note that a discount hyperparameter γ, common to gradient-based RL methods [333], is not required. This is since it is irrelevant in the maximally sparse reward setting. Finally, options for parallelization are outlined. In principle, nested parallelization is possible with an inner and outer parallelization of Steps 10-18 and 7-18, respectively. The former refers to N_{tasks} solutions for a given parameter vector θ_i, whereas the latter parallelizes n parameter perturbations. For final experiments, Steps 7-18 were implemented asynchronously.

Analysis

According to classifications in [119], TSHC is a gradient-free *instance-based* simulation optimization method, generating new candiate solutions based on only the current solution and *random search* in its neighborhood. Because of its hill climbing (greedy) characteristic, it differs from a) *evolutionary* (population-based) methods that construct solution by combining others typically using weighted averaging [371], [327], and b) from *model-based* methods that use *probability distributions* on the space of solution candidates, see [119] for a survey. In its high-level structure, Algorithm 4 can be related to the industrial strength COMPASS algorithm [375]. Within a global stage, they identify several possible regions with locally optimal solutions. Then, they find local optimal solutions for each of the identified regions. In a third stage, they select the best solution among all identified locally optimal solutions. In our setting, these regions are enforced as the separate training tasks and the best solution for all of these is selected.

An important role is held by σ_{pert}. In combination with sufficiently large n, it must be large enough to permit sufficient exploration such that a network parametrization solving all tasks can be found. In contrast, the effect of decreasing σ_{pert} with an increasing number of solved tasks is that, ideally, a speedup in learning progress results from the assignment of more of solution candidates θ_i closer in variance to a promising parametrization θ, see Step 8. For final experiments, $\sigma_{\text{pert}}^{\text{min}}$ was dismissed to not constrain σ_{pert}-adaptation.

Steps 25-27 are discussed. For the case that for a specific i_{iter}-iteration not all tasks have been solved, $i^\star = \arg\max_i \{N_i^{\text{tasks},\star}\}_{i=1}^n$ has been considered as an alternative criterion for Step 25. Several remarks can be made. First, Step 25 and the alternative are not equal. This is because, in general, different tasks are solved in a different number of time steps. However, the criteria are *approximately* equivalent for sparse rewards (since J_i accumulates constants according to (2.56)), and especially for large T^{max}. The core advantage of employing Step 25 in TSHC is that it can, if desired, also be used in combination with *rich* rewards to accelerate learnig

progress (if a suitable rich reward signal can be generated). In such a scenario, $i^\star$ according to Step 25 is updated towards most promisining $J_{i^\star}$, then representing the accumulated *rich* reward. Thus, in contrast to (2.56), a rich reward could be represented by a weighted sum of squared errors between state $z_t \in \mathbb{R}^{n_z}$ and a reference $z_t^{\text{ref}} \in \mathbb{R}^{n_z}$,

$$r_t = \begin{cases} -\infty, & \text{if } f_t^{\text{crash}} == 1, \\ -\sum_{l=0}^{n_z-1} \alpha_l (z_t(l) - z_t^{\text{ref}}(l))^2, & \text{otherwise}, \end{cases} \tag{2.61}$$

where α_l, are trade-off hyperparameters and scalar elements of vectors are indexed by l in brackets. Another advantage of the design in Algorithm 4 according to Step 25-27 is its *anytime solution* character. Even if not all N_{tasks} are solved, the solution returned for the tasks that *are* solved, typically is of good quality and optimized according to Steps 25-27.

If for all N_{tasks} tasks there exists a feasible solution for a given system model and a sufficiently expressive network structure parameterized by θ, then Algorithm 4 can find such parametrization for sufficiently large hyperparameters N_{restarts}, $N_{\text{iter}}^{\max}$, n, $T^{\max}$ and $\sigma_{\text{pert}}^{\max}$. The solution parametrization $\theta^\star$ is the result from the initialization Step 4 and parameter perturbations according to Step 8, both nested within multiple iterations. As noted in [187], for optimization via simulation, a global convergence guarantee provides little practical meaning other than reassuring a solution will be found *eventually* when simulation effort goes to infinity. The same reference states that a convergence property is most meaningful if it can help in designing suitable stopping criteria. Here, there are two conceptual levels of stopping criteria: first, the solution of all training tasks, and second, the refinement of solutions.

Control design is implemented hierarchically in two steps. First, suitable *training tasks* (desired motion primitives) are defined. Then, these are *encoded* in the network by the application of TSHC. This has practical implications. First, it encourages to train on *deterministic* tasks. Furthermore, at every i_{iter}, it is *simultaneously* trained on all of these separate tasks. This has the benefit that the best parametrization, $\theta^\star$, is clearly defined in Step 21, maximizing the accumulated P-measure over all tasks. Second, it enables to provide *certificates* on the learnt performance. The

certificates are given by provision of a) the employed vehicle model, and b) the list of encoded tasks (motion primitives). Note that such certificates cannot be given for the class of *stochastic* continuous action RL algorithms that are derived from the Stochastic Policy Gradient Theorem [350]. This class includes all stochastic actor-critic algorithms, including A3C [280] and PPO [333].

Discussion and Comparison with related RL Algorithms

A discussion of popular continuous control methods that use neural network for function approximation is given, focusing on one stochastic [333], one deterministic policy gradient method [253], and one evolution strategy [327]. The methods are discussed in relative detail to underline aspects of TSHC.

First, the *stochastic policy gradient* method PPO [333] is discussed. Suppose that a stochastic continus control vector is sampled from a Gaussian distribution parameterized[3] by θ such that $a_t \sim \pi(a_t|s_t, \theta)$. Then,

$$J(\theta) = \mathop{\mathrm{E}}_{s_t, a_t}\left[g(s_t, a_t \sim \pi(a_t|s_t, \theta))\right], \tag{2.62}$$

is defined as the expected accumulated and time-discounted reward when at state s_t drawing a_t, and following the stochastic policy for all subsequet times when acting in the simulation environment. Since function $g(s_t, a_t)$ is a priori not know, it is parameterized by $\theta_{V,\text{old}}$ and estimated. Using RL-terminology, in the PPO-setting, $g(s_t, a_t)$ represents the *advantage function*. Then, using the "log-likelihood trick", and subsequently a first-order Taylor approximation of $\log(\pi(a_t|s_t, \theta))$ around some reference $\pi(a_t|s_t, \theta_{\text{old}})$, the following parameterized cost function is obtained as an *approximation* of (2.62),

$$\tilde{J}(\theta) = \mathop{\mathrm{E}}_{s_t, a_t}\left[\hat{g}_t(\theta_{V,\text{old}})\frac{\pi(a_t|s_t, \theta)}{\pi(a_t|s_t, \theta_{\text{old}})}\right]. \tag{2.63}$$

[3]In this setting, mean and variance of the Gaussian distribution are the output of a neural network whose parameters are summarized by lumped θ.

Finally, (2.63) is modified to the final PPO-cost function [333]

$$\hat{J}(\theta) = \mathop{\mathrm{E}}_{s_t,a_t}\big[\min\Big(\hat{g}_t(\theta_{V,\text{old}})\frac{\pi(a_t|s_t,\theta)}{\pi(a_t|s_t,\theta_{\text{old}})}, \\ \text{clip}(\frac{\pi(a_t|s_t,\theta)}{\pi(a_t|s_t,\theta_{\text{old}})}, 1-\epsilon, 1+\epsilon)\hat{g}_t(\theta_{V,\text{old}})\Big)\big], \tag{2.64}$$

whereby the advantage function is estimated by the policy parameterized by $(\theta_{V,\text{old}}, \theta_{\text{old}})$, which is run for T consecutive time steps such that for all t the tuples $(s_t, a_t, r_t, s_{t+1}, \hat{g}_t(\theta_{V,\text{old}})$ can be added to a *replay buffer*, from which later minibatches are drawn. According to [333], the estimate is $\hat{g}_{T-1}(\theta_{V,\text{old}}) = \kappa_{T-1}$ with $\kappa_{T-1} = r_{T-1} + \gamma V(s_T, \theta_{V,\text{old}}) - V(s_{T-1}, \theta_{V,\text{old}})$, $\hat{g}_{T-2}(\theta_{V,\text{old}}) = \kappa_{T-2} + \gamma\lambda\hat{g}_{T-1}(\theta_{V,\text{old}})$ and so forth until $\hat{g}_0(\theta_{V,\text{old}})$, and where $V(s, \theta_{V,\text{old}})$ represents a *second*, the so-called critic neural network. Then, using uniform randomly drawn minibatches of size M, parameters (θ_V, θ) of both networks are updated according to
$\arg\min_{\theta_V} \frac{1}{M}\sum_{i=0}^{M-1}\left(V(s_i,\theta_V) - (\hat{g}_i(\theta_{V,\text{old}}) - V(s_i,\theta_{V,\text{old}}))\right)^2$ and
$\arg\max_{\theta} \frac{1}{M}\sum_{i=0}^{M-1}\mathcal{G}_i(\theta)$, with $\mathcal{G}_i(\theta)$ denoting the argument of the expectation in (2.64) evaluated at time-index i. This relatively detailed discussion is given to underline following observations. With first the introduction of a parameterized estimator, then a first-order Taylor approximation, and then clipping, (2.64) is an arguably crude approximation of the original problem (2.62). Second, the complexity with *two* actor and critic networks is noted. Typically, both are of the same dimensions apart from the output layers. Hence, when not sharing weights between the networks, approximately *twice* as many parameters are required. However, *when* sharing any weights between actor and critic network, then optimization function (2.64) must be extended accordingly, which introduces another approximation step. Third, note that gradients of both networks must be computed for backpropagation. Fourth, the dependence on *rich* reward signals is stressed. As long as the current policy does not find a solution candidate, in a *sparse* reward setting, all r_i are uniform. Hence, there is no information permitting to find a suitable parameter update direction and all of the computational expensive gradient computations are

essentially not usable[4]. Thus, the network parameters are still updated entirely at random. Moreover, *even if* a solution candidate trajectory was found, it is easily averaged out through the random minibatch update. This underlines the problematic of *sparse* rewards for PPO. Sixth, A3C [280] and PPO [333] are by nature *stochastic* policies, which *draw* their controls from a Gaussian distribution (for which mean and variance are the output of a trained network with current state as its input). Hence, exact repetition of any task (e.g., the navigation between two locations) cannot be guaranteed. It can only be guaranteed if dismissing the variance component and consequently using solely the mean for deterministic control. This can be done in practice, however, represents another approximation step.

Deterministic policy gradient method DDPG [253] is discussed. Suppose a deterministic continuous control vector parameterized such that $a_t = \mu(s_t, \theta)$. Then, the following cost function is defined,

$$J(\theta) = \mathop{\mathrm{E}}_{s_t, a_t}\left[g(s_t, a_t = \mu(s_t, \theta))\right] = \mathop{\mathrm{E}}_{s_t}\left[g(s_t, \mu(s_t, \theta))\right].$$

Its gradient can now be computed by simply applying the *chain-rule* for derivatives [341]. Introducing a parameterized estimate of $g(s_t, a_t)$, which here represents the *Q-function* or *action value function* (in contrast to the advantage function in above stochastic setting), the final DDPG-cost function [253] is

$$J(\theta) = \mathop{\mathrm{E}}_{s_t}\left[\hat{g}(s_t, \mu(s_t, \theta), \theta_Q)\right].$$

Then, critic and actor network parameters (θ_Q, θ) are updated as $\arg\min_{\theta_Q} \frac{1}{M}\sum_{i=0}^{M-1}(\hat{g}(s_i, a_i, \theta_Q) - (r_i + \gamma Q(s_{i+1}, \mu(s_{i+1}, \theta_{\text{old}}), \theta_{Q,\text{old}})))^2$ and $\arg\min_{\theta} \frac{1}{M}\sum_{i=0}^{M-1} Q(s_i, \mu(s_i, \theta), \theta_Q)$, where minibatches are used and with slowly tracking target network parameters $(\theta_{Q,\text{old}}, \theta_{\text{old}})$. Several remarks can be made. First, the Q-function is updated towards only its *one-step* ahead target. It is obvious that rewards are therefore propagated very slowly. For sparse rewards this is even more problematic than for rich

[4]It is mentioned that typically the first, for example, 50000 samples are collected *without* parameter update. However, even then that threshold must be selected, and the fundamental problem still perseveres.

rewards, especially because of the additional danger of averaging out important update directions though random minibatch sampling. Furthermore, and analogous to the stochastic setting, for the sparse reward setting, as long as no solution trajectory was found, all of the gradient computations are not usable and all network parameters are still updated entirely at random. DDPG is an *off-policy* algorithm. In [253], exploration of the simulation environment is achieved according to the current policy plus additive noise following an *Ornstein-Uhlenbeck* process. This is a mean-reverting linear stochastic differential equation [126]. A first-order Euler approximation thereof can be expressed as the action exploration rule $a_t = \mu(s_t, \theta)(1 - P_\theta^{\text{OU}}) + P_\sigma^{\text{OU}}\epsilon,\ \epsilon \sim \mathcal{N}(0, I)$, with hyperparameters $(P_\theta^{\text{OU}}, P_\sigma^{\text{OU}}) = (0.15, 0.2)$ in [253]. This detail is provided to stress a key difference between policy gradient methods (both stochastic and deterministic), and methods such as [327] and TSHC. Namely, while the former methods sample controls from the stochastic policy or according to *heuristic* exploration noise before updating parameters using minibatches of *incremental* tuples (s_i, a_i, r_i, s_{i+1}) plus $\hat{g}_i(\theta_{V,\text{old}})$ for PPO, the latter directly work in the *parameter space* via local perturbations, see Step 8 of Algorithm 4. This approach appears particularly suitable when dealing with sparse rewards. As outlined above, in such setting, parameter updates according to policy gradient methods are also entirely at random, however, with the computationally significant difference of first an approximately four times as large parameter space and, second, the unnecessary costly solution of non-convex optimization problems as long as no solution trajectory has been found. A well-known issue in training neural networks is the problem of vanishing or exploding gradients. It is particularly relevant for networks with saturating nonlinearities and can be addressed by recent batch [197] and layer normalization [15]. In both normalization approaches, *additional* parameters are introduced to the network which must be learnt (bias and gains). Said issue is not relevant for the proposed gradient-free approach, in which also saturating activation functions are used, see Section 2.2.4.

This work is originally inspired by and most closely related to [327]. The main differences are discussed. The latter evolutionary (population-

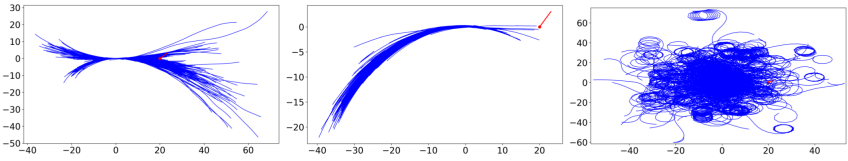


Figure 29: Experiment 1. 1000 training trajectories resulting from the application DDPG (Left), PPO (Middle) and TSHC (Right), respectively. The effect of virtual constraints on velocity is particularly visible for DDPG. For the given hyperparameter setting ([333, Table 3] and $T_{\max} = 100$), the trajectories for PPO have little spread and are favoring reverse driving. TSHC has a much better exploration strategy resulting from noise perturbations in the parameter space. The task is solved by TSHC according to Figure 30.

based) strategy updates parameters using a *stochastic gradient estimate*. Thus, it updates $\theta \leftarrow \theta + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} R_i \zeta_i$, where hyperparameters α and σ denote the learning rate and noise standard deviation, and where R_i here indicates the stochastic scalar return provided by the simulation environment. This weighted averaging approach for the stochastic gradient estimate is not suitable for our control design method when using separate deterministic training tasks in combination with maximally sparse rewards. Here, hill climbing is more appropriate. This is since most of the n trajectory candidates do not end up at z^{goal} and are therefore not useful. Note also that only the introduction of virtual constraints on velocity permitted us to quickly train with maximally sparse rewards. It is well known that for gradient-based training, especially of RNNs, the learning rate (α in [327]) is a critical hyperparameter choice. In the hill climbing setting this issue does not occur. Likewise, *fitness shaping* [371], also used in [327], is not required. Note that above σ has the same role as σ_{pert}. Except, in our setting, it additionally is adaptive according to Steps 30 and 31 in Algorithm 4. As implemented, this is only possible when training on multiple separate tasks. Other differences include the parallelization method in [327], where random seeds shared among workers permit each worker to only need to send and receive the scalar return of an episode to and from each other worker. All perturbations and param-

Table 2: Experiment 1. Number of scalar parameters (weights) that need to be identified for DDPG, PPO and TSHC, respectively. TSHC requires to identify the least by a large margin. In general, roughly by a factor 4. The fact that PPO here requires *exactly* four times the number of parameters of TSHC is a special case for $n_a = 2$ (not generalizable for arbitrary n_a).

	DDPG	PPO	TSHC
N_{var}	19078	18440	4610

Table 3: Experiment 2a. Learning 5 tasks. Comparison of six architectures A1-A6. Hyperparameters are selected as $n = 2000$, $T_{\max} = 100$, and terminating upon the first solution found (no refinement step, no additional restart). Results are average over 5 runs. The total number of parameters to be identified, the number of iterations until the first solution solving all tasks, the total accumulated pathlength and solve time for Algorithm 4 are denoted by N_{var}, $\bar{N}^{\star}_{\text{iter}}$, $\bar{P}^{\star}$ and $\bar{T}^{\star}$, respectively.

	A1	A2	A3	A4	A5	A6
N_{var}	4610	4610	118	166	169	12930
$N^{\star}_{\text{iter}}$	19	6	5	5	5	13
$\bar{P}^{\star}$	152.7	152.6	150.51	148.31	152.4	157.6
$\bar{T}^{\star}$	481.1	141.3	144.9	165.95	110.6	367.8

eters are then reconstructed locally by each worker (for n workers there are n reconstructions at each parameter-iteration step). This requires precise control of each worker and can in rare cases lead to differing CPU utilizations among workers due to differing episode lengths. Therefore, they use a capping strategy on maximal episode length. Our proposed method is less sophisticated with *one* synchronized parameter update, which is then sent to all workers.

2.2.5 Numerical Experiments

This section highlights different aspects of Algorithm 4. For all experiments, both continuous steering and velocity control are required. Throughout, tolerances are set as $(\epsilon_d, \epsilon_\psi, \epsilon_v) = (1, 10\pi/180, 10/3.6)$.

In Experiment 1, for the implementation of the two policy gradient methods, Tensorflow (without GPU-support) was used. All experiments were conducted on a laptop running Ubuntu 16.04 equipped with an

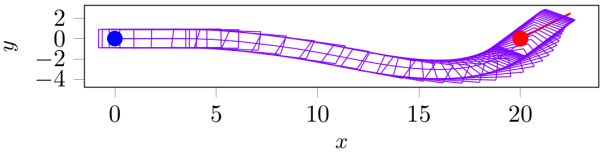


Figure 30: Experiment 1. The network controller trained by the TSHC method solves the task in only 2.1 seconds, when terminating upon the first solution found (no refinement step, no additional restart). Rectangular vehicle dimensions are displayed. Blue and red ball and indicators visualize z_0 and z^{goal}.

Intel Core i7 CPU @2.80GHz×8, 15.6GB of memory, and using Python 2.7. For the implementation of Algorithm 4, the only libraries employed are Python's `numpy` and `multiprocessing`.

Experiment 1: Comparison with Policy Gradient Methods

To underline conceptual differences between TSHC and the two policy gradient methods DDPG [253] and PPO [333], a freeform navigation task with $z_0 = [0, 0, 0, 0]$ and $z^{\text{goal}} = [20, 0, \pi/4, 0]$ was considered. The same network architecture from [333] is used: a fully-connected MLP with two hidden layers of 64 units before the output layer. Eventhough this is the basic setup, considerable differences between DDPG, PPO and TSHC are implied. First, both DDPG and PPO are each composed of a total of four networks: one actor, one critic, one actor target and one critic target network. For DDPG, further parameters result from batch normalization [197]. The number of parameters N_{var} that need to be identified are indicated in Table 2. To enable a fair comparison, all of DPPG, PPO and TSHC are permitted to train on 1000 full rollouts according to their methods, whereby each rollout lasts at most $T_{\text{max}} = 100$ timesteps. Thus, for TSHC, $N_{\text{restarts}} = 1$ and $n = 1000$ are set. For both PPO and DDPG, this implies 1000 iterations. The results are summarized in Figure 29 and 30. The following observations can be made. First, in comparison to TSHC, for both DDPG and PPO significantly more parameters need to be identified. Second, DDPG and PPO do *not* solve the task based on 1000 training simulations. In contrast, as Figure 29 demonstrates, TSHC

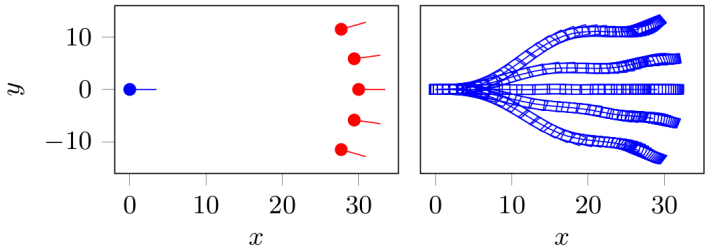


Figure 31: Experiment 2a. Learning 5 tasks. (Left) Problem formulation with start (blue) and end (red) positions as well as headings. Start and desired end velocities are zero. (Right) LSTM-solution after less than 150s training time. Vehicle dimensions are displayed.

has a much better exploration strategy resulting from noise perturbations in *parameter space* and also solves the task in just 2.1s. Finally, note that no σ_{pert}-iteration is conducted. It is not applicable since a *single* task is solved with an initial $\sigma_{\text{pert}}^{\text{max}} = 10$. Because of these findings (other target poses were also tested with qualitatively equivalent results) and above discussion about the handling of *sparse rewards* for related RL work and the fact that DDPG and PPO have no useful gradient direction for their parameter update or may average these out through random minibatch sampling, the focus in the subsequent sections is on TSHC and its analysis.

Experiment 2: Zone Navigation without Obstacles

Various zone navigation experiments without obstacles are conducted. Therefore, to eliminate the influence of the I-network park, NETW$_{sI}$ and NETW$_{\tilde{s}}$ here conincide.

For Experiment 2a, six different architectures (A1-A6) are compared. Results are summarized in Table 3 and Figure 31. For A1 and A2, the same architecture from [333] is considered: a fully-connected MLP with two hidden layers of 64 units before the output layer. A1 and A2 solely differ by the absence and inclusion of *adaptive* σ_{pert} according to Algorithm 4. For A3 and A4, a single GRU- and LSTM-cell before the output layer are considered, respectively. For A5, a fully-connected MLP with

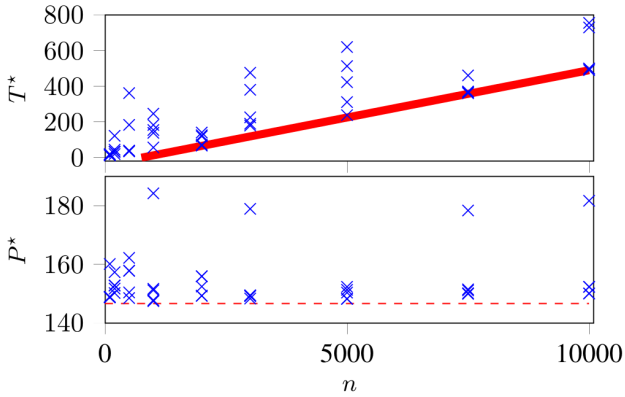


Figure 32: Experiment 2b. The results for A2 as a function of different n are displayed. The problem setup is as in Figure 31. For each n, five restarts are conducted. For each restart, the i_{iter}-loop is terminated upon the first solution found (no refinement step). The solid red line indicates the interpolated lower bound on solve times. The fine red dashed line denotes the minimum accumulated pathlength (146.6m) after $N_{\text{iter}} = 100$ with refinement step.

two hidden layers of 10 and 9 units before the output layer is used, such that approximately the same number of parameters as for A4 are identified. For A6, a MLP with four hidden layers of 64 units before the output layer is considered. Several observations can be made. First, as the comparison of A1-A2 solve times shows, adaptive σ_{pert} is useful in providing speedup. Due to adaptive σ_{pert}, the A6-network which is almost 3-times larger than A1, can encode the five tasks 23.6% faster than A1, which does not adapt σ_{pert}. Second, single GRU- and LSTM-cells as well as a smaller MLP (A5) with few parameters can encode the tasks, too.

For Experiment 2b, the effect of n is analyzed for A2, see Figure 32. As expected, solve times clearly increase with increasing n. However, solution quality (measured by total accumulated pathlength) is not affected in the same manner. Suboptimal solutions typically include an unnecessary pathlength-prolonging circular turn before reaching a goal position. Importantly, large n do not automatically imply best results. Often a large spread in $T^\star$ and $P^\star$ over different restarts could be observed. This underlines the importance of multiple restarts and refinement steps for

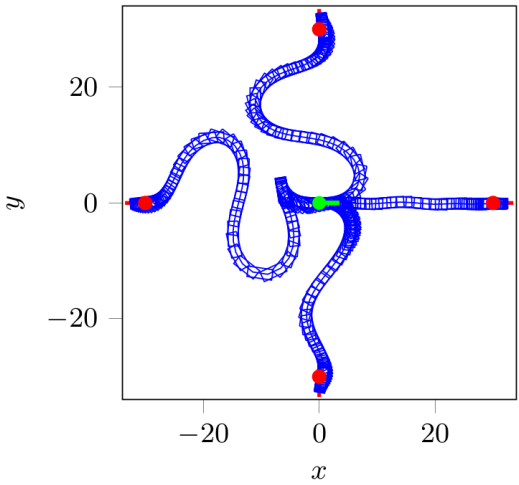


Figure 33: Experiment 2c. Learning 4 tasks. Start (green) and end (red) positions as well as headings are displayed. The latter are distributed symmetrically along a circle of radius 30m. Start and desired end velocities are zero. The LSTM-solution (A4-architecture with $N_{\text{var}} = 166$) after 435.8s trainings time and $N^{\star}_{\text{iter}} = 16$ is visualized. Note that one task (with $x^{\text{goal}} = -30$) is solved by driving *reverse* for some part. This is enabled by system modeling according to Section 2.2.3. Since not conducting any restarts and refinement steps, overall suboptimal (non-minimum accumulated pathlength) trajectories result.

optimized solutions.

For Experiment 2c, A4 (a single LSTM-cell before the output layer) is used for the learning of 4 tasks with goal positions distributed symmetrically on a circle, see Figure 33.

For Experiment 2d, see Figure 34. The importance of conducting multiple restarts is underlined. Not always all tasks could be solved for every restart.

Experiment 3: Navigation with an Obstacle

The following problem is addressed: overtaking of an obstacle if it exists, and straight road following if there is none. Therefore, two tasks are formulated: one with and one without any obstacle. A relative velocity

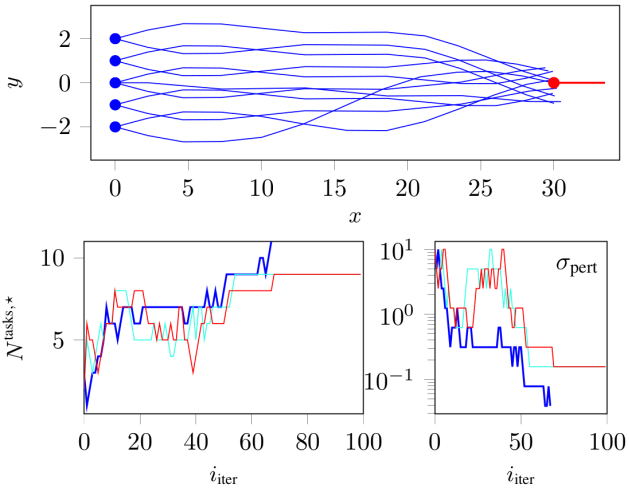
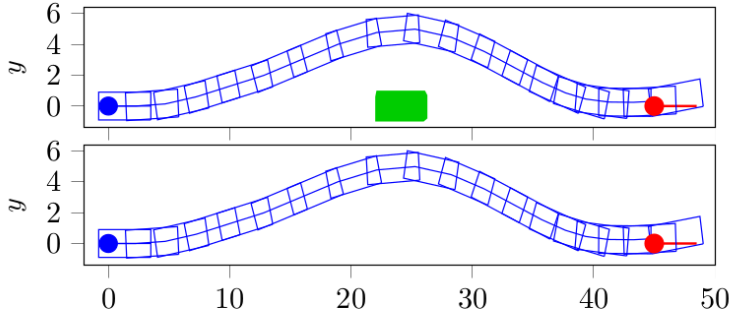


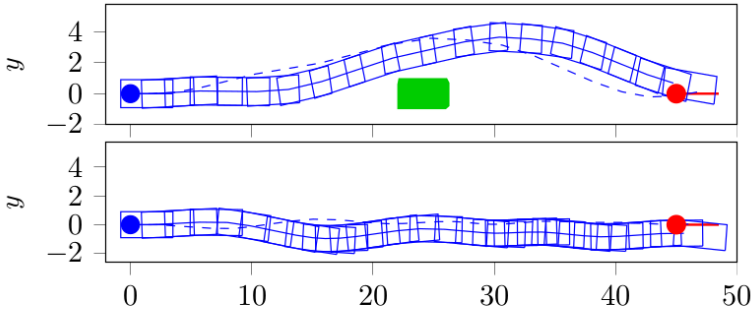
Figure 34: Experiment 2d. Learning 11 tasks. Architecture A2 is employed. One out of three restarts solved all tasks within $N_{\text{iter}}^{\max} = 100$ for each restart and $n = 100$. No refinement steps are conducted, which explains the asymmetry of trajectories. The total training time was 620.1s.

difference between an ego and a lead car of 30km/h is assumed, and $z_0 = [0, 0, 0, 30/3.6]$ and $z^{\text{goal}} = [45, 0, 0, 30/3.6]$ are set. A front range-view of 20m is selected such that late obstacle detection is simulated. A rear range-view of 10m is assumed. The obstacle is located at $x = 22$. Thus, it is initially not seen. Pixel resolution is set to 0.5m/pixel: 60 and 20 pixels over the entire range view length of 30m and width of 10m, respectively. See also Figure 25.

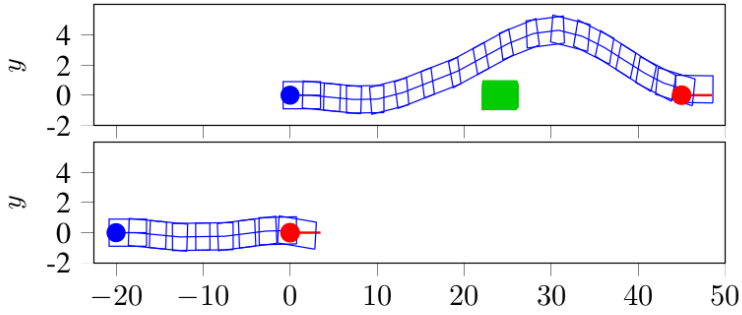
First, NETW$_{sI}$ according Figure 26 is trained using FC-layers with 64 units for each of S-, I- and C-part, and additionally $\Gamma = 1$. Thus, input dimensions are $n_s(1 + \Gamma) = 8$ and $n_I(1 + \Gamma) = 80$, and $N_{\text{var}} = 14146$. A typical learning result is displayed in Figure 35(a). This result was not expected. While the minimum accumulated pathlength criterion theoretically guarantees a global optimum (by nonnegativity of pathlength and a single obstacle guaranteeing trajectory feasibility), it turned out to be very difficult to escape local optima. The trajectory solving the more complex OA-task automatically also solves the simpler task without ob-



(a) Method M1. Naive training of NETW$_{sI}$.

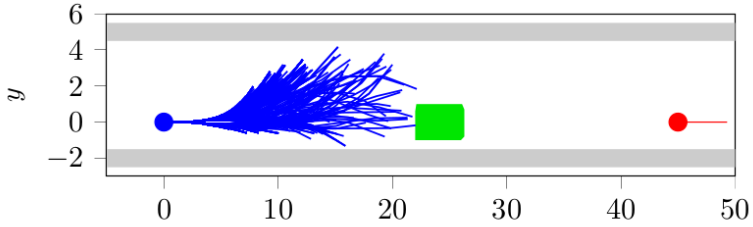


(b) Method M2. Hierarchical training with *rich* rewards in the second stage. Dashed lines indicate references obtained in the first stage.

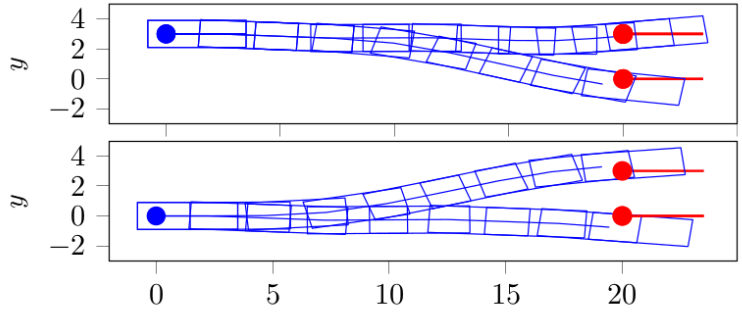


(c) Method M3. Training of NETW$_{sI}$

Figure 35: Experiment 3. For better illustration, obstacle avoidance and road following task are separated into two subplots. [Figure continued on the next page].



(d) Method M4. Introduction of road bounds (gray). Display of a training step with $n = 1000$ exploration trajectories (without vehicle dimensions). Upon any crash with either a road bound or the obstacle, the time-rollout is interrupted.



(e) Method M5. Learning 4 tasks according to NETW$_{\tilde{s}}$. Only for clarity, the trajectories are partitioned into two plots. The LSTM-architecture A4 ($N_{\text{var}} = 166$) is employed. After 215.3s training time and $N^{\star}_{\text{iter}} = 7$, the four tasks are learnt (no additional restart or refinement step).

Figure 36: Experiment 3. [Continued from previous page].

stacle. Qualitatively identical behavior was observed when replacing FC-layers by LSTM-cells. Note that when replacing both S- and I-part by LSTM-cells, 58154 parameters need to be identified.

Therefore, an alternative two-step method was tested:

1. Solve each task *isolatedly* by Algorithm 4 (with sparse rewards), and store the resulting trajectories as $z_t^{\text{ref},j_{\text{task}}},\ \forall t = 0,\dots,T^{j_{\text{task}}},\ \forall j_{\text{task}} = 1,\dots,N_{\text{tasks}}$.

2. Solve another RL-problem by Algorithm 4, now simultaneously treating *all* tasks and using *rich* rewards according to (2.61), with $z_t^{\text{ref},j_{\text{task}}}$ from Step 1 serving as references to obtain $r_t^{j_{\text{task}}},\ \forall t = 0,\dots,T^{j_{\text{task}}},\ \forall j_{\text{task}} = 1,\dots,N_{\text{tasks}}$.

Variants are a) to also include controls as references in the rich reward signal, and b) to not terminate the t-iteration upon exceeding $T^{j_{\text{task}}}$, but instead maintaining $z_{T^{j_{\text{task}}}}^{\text{ref},j_{\text{task}}}(l)$ as reference, $\forall t > T^{j_{\text{task}}}$. This two-step method appears appealing because it permits to first optimize each task isolatedly to a desired performance. However, it has two caveats. First, it requires to solve $(N_{\text{tasks}}+1)$ RL-problems, which is time-consuming. Second, the time-reference tracking over different tasks is difficult. As Figure 35(b) illustrates, both tasks are learnt, however, the second task trajectory (when not encountering any obstacle) exhibits suboptimal wiggles. The third and most important caveat is that it requires to set hyperparameters $\alpha_l,\ \forall l$, to trade-off different *units* (meters, radians, and m/s) within the state vector. These were found to be very influential. For the result in Figure 35(b), $\alpha_l = 1$ for $l = 0,1$ and $\alpha_l = 0$ for $l = 2,3$ are set, thereby only considering x, y-coordinates, both measured in meters. This setting, however, turned out to not be sufficient to solve tasks with major steering actuation as illustrated in Figure 33. Conducting various experiments with above method further consolidated our preference for sparse reward signals. This is since for maximally sparse rewards, different units within the state vector are irrelevant.

For a third method, the original tasks are reformulated by recalibrating (z_0, z^{goal})-combinations for the case with and without obstacle, see Figure 35(c). For the second task, $x_0 = -20$ and $z^{\text{goal}} = 0$ are set in regard

of the front range-view of 20m. After the (z_0, z^{goal})-resetting, the same training technique from method M1 was applied (with $N_{\text{var}} = 14146$). Now, both tasks are learned, and the original problem formulated at the beginning of this section is solved by resetting (z_0, z^{goal})-combinations upon first obstacle detection.

Fourth, the influence of road bounds on learning progress was analyzed. In addition, any exploration trajectory is dismissed as soon $v_t < 0$ or $x_t < 0$. Figure 36(d) visualizes the effect for $n = 1000$ after $i_{\text{iter}} = 5$. Note that because of different velocities, the pixel-discretization resolution, and discretization of obstacle edges crashes are detected irregularly. Because of the many crashes and consequent discarding as potential candidates for the next training iteration, it may (and frequently did) happen that *none* of the n candidates can be used for a parameter update for current i_{iter}. Hence, learning progress is significantly hindered.

Fifth, motivated by the last observations, $\text{NETW}_{\tilde{s}}$ was revised for the problem. In fact, the problem can be solved by a) sequentially conducting a lane change, overtaking the obstacle in parallel, before conducting a second lane change if there exists an obstacle, and b) just maintaining the current lane in case there is no obstacle. Therefore, 4 training tasks can be defined. The front range view horizon of 20m is used as reference for the lane change, implying a lane-change upon the first possible obstacle detection. Figure 36(e) illustrates the learning results. Note that in general only 3 task formulations, or even only two when permitting control mirroring, can be sufficient for problem solution.

Inverted Pendulum

The discussion of tolerance levels in Section 2.2.4 motivated to consider an alternative method for tasks requiring stabilization. An analogy to optimal control is drawn. In linear finite horizon MPC, closed-loop stability can be guaranteed through a *terminal state constraint set* which is invariant for a *terminal controller*, often a linear quadratic regulator (LQR), see [270]. In a RL setting, the following procedure was considered. First, design a LQR for stabilization. Second, compute the *region of attraction* of the LQR controller [353, Sect. 3.1.1]. Third, this region of attraction

can now be used as stopping criterion, *replacing* the heuristic ϵ-tolerance selection.

For evaluation, the inverted pendulum system equations and parameters from [10] were adopted (four states, one input). However, in contrast to [10], which assumes just two discrete actions (maximum and minimum actuation force), here a *continuous* control variable is assumed which is limited by the two bounds, respectively. There are two basic problems: stabilization in the upright position with initial state in the same position, as well as a swing-up from the hanging position plus consequent stabilization in the upright position. For the application of TSHC, $(N_{\text{restarts}}, N_{\text{iter}}^{\max}, n, T^{\max}, \beta, \sigma_{\text{pert}}^{\max}) = (3, 100, 100, 500, 2, 10)$ are set, and the A2-architecture is used. The following remarks can be made. First, the swing-up plus stabilization task was solved in $T^{\star} = 43.5$s runtime of TSHC (without refinement step) and using sparse rewards (only obtained in the upright position $\pm 12^{\circ}$). For all three restarts a valid solution was generated. Note that $T^{\max} = 500$ in combination with a sampling time [10] of 0.02s corresponds to 10s simulation time. Stabilization in the upright position was achieved from 2.9s on. Rich reward signals were also tested, exploiting the deviation from current to goal angle as measure. However, they did not accelerate learning.

In another experiment, the objective was to *simultaneously* encode the following *two* tasks in the network: stabilization in the upright position with initial state in the same position, and a swing-up from the hanging position plus consequent stabilization in the upright position. The runtime of TSHC (without refinement step) was $T^{\star} = 264.4$s, with 2 of 3 restarts returning a valid solution and using sparse rewards. Instead of learning both tasks simultaneously according to TSHC, it was also attempted to learn them by selecting one of the two tasks at random at every i_{iter}, and consequently conducting Steps 6-34. Since the two tasks are quite different, this procedure could not encode a solution for *both* tasks. This is mentioned to exemplify the importance of training *simultaneously* on separate tasks, rather than training on a single tasks with (z_0, z^{goal}) combinatons varying over i_{iter}.

Finally, for system parameters [10], it was observed that the contin-

uous control signal was mostly operating at saturated actuation bounds (switching in-between). This is mentioned for two reasons. First, aforementioned LQR-strategy could therefore never be applied since LQR assumes absence of state and input constraints. Second, it exemplifies the ease of RL-workflow. A combination of network parametrization, TSHC, and a discretization scheme (Euler or Runge-Kutta) enables quick nonlinear control design, even without system insights.

Discussion of Experiments and Summary of Findings

First, for automotive applications our preferred network architecture clearly is NETW$_{\tilde{s}}$. Training the alternative NETW$_{sI}$ has several disadvantages: a much larger network sizes due to the I-part, slow learning progress in constrained environments and the difficulty in shaping desired learning results. In contrast, NETW$_{\tilde{s}}$ permits fast learning due to perturbations in the parameter space in combination with obstacle-free freeform navigation tasks, which for sufficiently large T^{max} naturally guarantee feasibility of solution trajectories. Then, for obstacle avoidance, receding horizon setpoint setting becomes crucial, using I_t as a filter to reset z^{goal}, for example, upon predicted obstacle crash to account for vehicle dimensions or for adaptation of v^{goal} based on, e.g., detected road surface.

Second, the employment of *sparse rewards* for training in combination with *virtual constraints* in setpoint-proximity has several advantages. It naturally avoids the need to introduce trade-off hyperparameters for the weighting of states in different units. And, it permits a solution trajectory between z_0 and z^{goal} to naturally evolve without biasing it by provision of a rich reference to track.

Third, several guidelines for the four key hyperparameters n, β, N_{restarts} and $N_{\text{iter}}^{\text{max}}$) were identified. Foremostly, the importance of multiple restarts is highlighted. Second, n is the most influential hyperparameter. It was found that huge n do not automatically translate to optimal solutions, but always significantly prolong learning duration. For both learning time and solution quality, it is recommended to start with small n in combination with a higher $N_{\text{iter}}^{\text{max}}$ to enable multiple iterations over parameters, and to only incrementally increase n when not all tasks could

be solved. By $\beta > 1$ it can be controlled how much σ_{pert} is adapted between two parameter iterations (the smaller the more conservative). Testing $\beta = 2$ and $\beta = 1.3$, it was found that the more aggressive $\beta = 2$ always outperformed. In general, adaptive σ_{pert} was found to clearly improve learning speed. Note that our logic for σ_{pert} adaptation differs from [371, Alg. 5 and 6], where also adaptive σ_{pert} is used, however, based on a stochastic gradient estimate.

Fourth, the discussion on ϵ-levels is also related to the decision on system modeling in either the absolute or road-aligned coordinate system. In contrast to the *spatial-based* LSV-MPC method from Section 2.1, for the neural approach for two reasons the absolute coordinate framework is clearly favored. First, in contrast to an online-solving approach, spatial parameterization is now not needed for the avoidance of reference velocity-related issues [148]. This is since training tasks are encoded *offline*. Second, the road-aligned coordinate framework is always created online during road-following. It is thus much more difficult to train offline for varying road shapes, rather than to train on setpoints in an absolute coordinate frame.

2.2.6 Hierarchical Controller Parametrization

For single-vehicle motion planning by neural network control, the preferred control parametrization consists of three hierarchies as visualized in Figure 37.

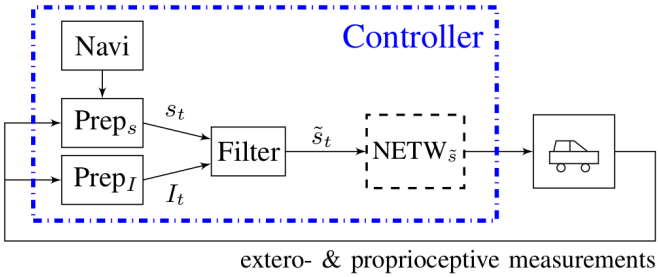


Figure 37: Hierarchical controller parametrization for the neural network approach to motion planning of automated vehicles.

2.3 Multi-vehicle Motion Planning

This section summarizes [154]:

- M. Graf Plessen, D. Bernardini, H. Esen, and A. Bemporad, "Multi-automated vehicle coordination using decoupled prioritized path planning for multi-lane one- and bi-directional traffic flow control," in *IEEE Conference on Decision and Control*, pp. 1582-1588, 2016.

A method for the coordination of multiple automated vehicles using priority schemes for decoupled motion planning for multi-lane one- and bi-directional traffic flow control is presented. The focus is on tube-like roads and non-zero velocities (no complete standstill maneuvers). We assume inter-vehicular communication (car-2-car) and a centralized or decentralized coordination service. We distinguish between different driving modes including adaptive cruise control (ACC) and obstacle avoidance (OA) for the handling of dynamic driving situations. We further assume that any controllable vehicle is equipped with proprioceptive and exteroceptive sensors for environment perception within a particular range field. In case of failure of the inter-vehicle communication system, the controllable vehicles can act as autonomous vehicles. The motivation is the control of a) one-directional multi-lane roads available for automated as well as unautomated objects with potentially, but not necessarily, varying reference speeds, and b) bi-directional traffic flow control making use of all available lanes, allowing, in general, object- and direction-wise variable reference speeds. For the one-directional case, we discuss a suitable deterministic priority scheme for throughput maximization and quickly reaching of a platooning state. For the bi-directional scenario, we derive a binary integer linear program (BILP) for the assignment of lanes to one of the two road traversal directions that can be solved optimally via linear programming (LP). The approach is evaluated on three numerical simulation scenarios.

2.3.1 Introduction

With the advent of autonomous driving, a natural extension is the coordination of multiple automated vehicles. Indeed, car-2-car and car-2-infrastructure communication and subsequent real-time coordination is perceived to be the main enabling technology for maximized road safety, lane throughput, and congestion avoidance due to its *anticipative* nature and potential for deterministic traffic flow planning [338]. Fundamental for intelligent transportation systems (ITS) is inter-vehicle communication (IVC) [323]; for two surveys on the topic, see [250] and [339]. Car-2-car communication for coordinated driving bears multiple technological challenges, not the least to satisfy stringent real-time constraints [66], [30]. For the remainder, we assume car-2-car communication to be available in combination with a coordination service, see Figure 38.

For the coordination of multi-robot systems, there exist two main approaches, *centralized* and *decoupled* motion planning [302]. The time complexity for centralized approaches is exponential in the dimension of the combined configuration space of all individual robots [45]; thereby making it, in general, unsuitable for real-time robot motion coordination. In contrast, decoupled methods trade off optimality and completeness (they may fail to find a solution even if one exists) for computational efficiency. Here, a prioritized and decoupled method is employed. This comprises a priority sequence for the motion planning of the vehicle: the path of vehicle i is computed taking motion information of the prioritized previous $i-1$ vehicles into account. The avoidance of one or multiple obstacles results, in general, in a non-convex combinatorial optimization problem, since the set of possible safe trajectories around an obstacle is combinatorial (passing it from left or right). A rigorous, but currently untractable approach for real-time feasibility in automotive applications is the formulation of mixed integer quadratic or linear programs [330]. One approach to tackle the traffic congestion issue is platooning, i.e., the (longitudinal) coordination of multiple vehicles with small inter-vehicle gaps as an application of cooperative adaptive cruise control (CAAC) [170], [338]. The other main intelligent vehicle (IV)-based traffic manage-

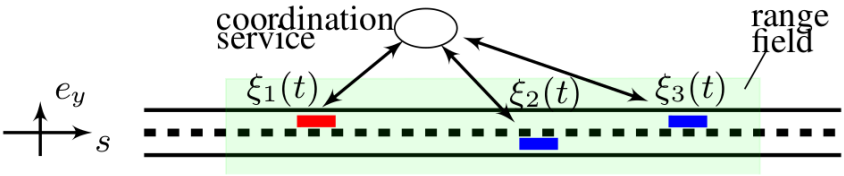


Figure 38: Illustration of car-2-car communication and coordination of automated vehicles via a service. The service receives state information of a group of vehicles located within a particular range field or segment of a road, computes a priority scheme and corresponding reference trajectories for each vehicle, before broadcasting. The service-computation may be conducted locally by one of the vehicles, e.g., by a leader as indicated by the red vehicle in a (wireless) self-organising vehicular ad hoc network (VANET), or, by an independent entity such as a web-service. For a detailed survey on inter-vehicle communication (IVC) systems, see [339].

ment approach is founded on self-organization performing maneuvers locally in a cooperative fashion [26]. In most countries traffic is organized laterally within speed lanes for more predictable movements of other vehicles. However, some countries, e.g., India, allow unorganized traffic where vehicles may travel anywhere inside road boundaries at arbitrary traveling speed, resulting in higher traffic bandwidth and significantly more overtaking [203]. In [202], elastic strips are used for the planning of autonomous vehicles in nonlane based one-directional unorganized traffic with absent speed limitations.

The presented method is intended to cover both organized and unorganized traffic in a natural manner, differing only by inputting a different reference velocity, e.g., for passenger cars, trucks, scooters or auto rickshaws, potentially depending on road speed limits or ground constitution. In addition to one-directional traffic flow control with the two distinct objectives of first, quickly reaching a platooning state, and second, throughput maximization allowing different vehicles to travel at variable reference velocities, we also discuss a method for traffic organization for the multi-lane *bi-directional* case. To the best of the authors' knowledge this is the first approach of this type.

2.3.2 Cooperative Driving System

We assume that within a particular range field, in addition to uncontrollable static or dynamic obstacles, there are multiple automated vehicles, each one equipped with proprioceptive and exteroceptive sensors, a path planning module, a reference tracking controller and an inter-vehicle communication system. Then, at every sampling instant T_s, Algorithm 5 is executed by the cooperative driving system.

Algorithm 5: Multi-automated vehicle coordination

1 Automated vehicles state estimation:
   - proprioceptive and exteroceptive state estimation.
   - implemented by sensor fusion of map databases, sensor measurements and dynamic vehicle models.

2 Communication:
   - state estimation information sent to a coordination service.

3 Coordination service:
   - computation of a priority scheme Π.
   - usage of a path planning module and driving mode selector for computation of constraints and reference trajectories for *each* of the automated vehicles within the given range field.

4 Broadcasting:
   - reference trajectory and constraint assignments to automated vehicles.

5 Automated vehicles tracking:
   - tracking of assigned reference trajectories under consideration of assigned constraints.

For the realization of cooperative driving, besides inter-vehicular communication, the control system onboard an automated vehicle must at least offer capabilities for: a) adaptive cruise control (ACC), b) path planning for multi-obstacle avoidance (OA), c) curved road-profile tracking (RT) and d) controlled braking (Brake). In [155] we presented such a control method. Note that the objectives of ACC with its distance keeping capabilities and OA with its characteristic approaching and overtaking

of objects are by definition conflicting. This motivated to distinguish between different driving modes including ACC, OA, RT and braking. For all of the following we assume:

Assumption 1. *Every automated vehicle can act autonomously and is equipped with capabilities for ACC, OA, RT and controlled braking.*

Randomized vs. Deterministic Priority Schemes

In [45], a method was presented to optimize priority schemes for decoupled prioritized path planning of a team of very slowly moving robots in mostly map-known indoor office-like environments using a *randomized* search with hill-climbing to minimize overall path length. This work was extended in [46] to a priori discard some priority sequences leading to infeasibility. Our automotive application with intermediate to high velocity range (30-130km/h) and tube-like roads differs significantly from the aforementioned scenarios. Slow and approximately constant velocities allow for good predictability. With increasing dynamic behavior trajectory predictions become complex and computationally expensive. In combination with stringent real-time requirements, this motivated us to focus on the design of deterministic priority schemes for both one- and bi-directional traffic flow control.

One-directional Traffic Flow

Let a *tube-like road* be characterized by variable s denoting the distance along the road centerline and e_y the lateral position with respect to the road centerline, see Figure 38. Space-varying road boundaries can be described by $e_y^{\max}(s) = 0.5e_y^w(s)$ and $e_y^{\min}(s) = -0.5e_y^w(s)$, where $e_y^w(s)$ is the road width at position s. A *nonlane road* is described by just its road boundaries. Let a subset of all states of automated vehicles $i = 1, \ldots, N^{\text{obj}}$ be denoted by $\xi_i(t) = [s_i(t),\ e_{y,i}(t),\ v_i(t)]$, where $s_i(t)$, $e_{y,i}(t)$ and $v_i(t)$ are the center of gravity (CoG)-position along the road centerline, $e_{y,i}(t)$ the lateral displacement and $v_i(t)$ the projected vehicle velocity along the road centerline at time $t \geq 0$, respectively. We define

$\xi_{ij}(t) = \xi_i(t) - \xi_j(t)$, and abbreviate $d_{ij}(t) = s_i(t) - s_j(t)$, $\Delta e_{y,ij}(t) = e_{y,i}(t) - e_{y,j}(t)$ and $\Delta v_{ij}(t) = v_i(t) - v_j(t)$.

Definition 1. *Let a priority scheme denote a sorted list of N^{obj} automated vehicles, $\Pi \in \mathbb{Z}_{++}^{N^{obj}}$, with $\Pi_i \in \{1, \dots, N^{obj}\}$ such that $\Pi_i \neq \Pi_j,\ \forall i, j = 1, \dots, N^{obj}$. The list is sorted such that the first element, Π_1, has the highest and $\Pi_{N^{obj}}$ the lowest priority order. The priority sequence implies that vehicle Π_i takes for its corridor and trajectory planning the reference trajectories of the prioritized vehicles Π_1 until Π_{i-1} into account.*

Objective 1. *We define the maximization of throughput by*

$$\min \int_0^T \sum_{i=1}^{N^{obj}} q_i(t)(v_i(t) - v_i^{ref}(t))^2 dt,$$

with T a time horizon, $v_i^{ref}(t)$ a reference velocity for vehicle i and $q_i(t)$ a weight.

Objective 2. *We define the objective of quickly reaching a platooning state as*

$$\begin{aligned}
\min\ & T && \text{(2.65a)}\\
s.t.\ & |d_{\Pi_i\Pi_j}(T) - d^{ref}_{\Pi_i\Pi_j}| < \epsilon_d, && \text{(2.65b)}\\
& |\Delta e_{y,\Pi_i\Pi_j}(T) - \Delta e^{ref}_{y,\Pi_i\Pi_j}| < \epsilon_{\Delta e_y}, && \text{(2.65c)}\\
& |\Delta v_{\Pi_i\Pi_j}(T)| < \epsilon_v, && \text{(2.65d)}\\
& j = i+1,\ i, j \in \{1, \dots, N^{obj}\}, && \text{(2.65e)}\\
& \Pi \in \mathbb{Z}_{++}^{N^{obj}},\ \Pi_i \neq \Pi_j,\ \forall i, j = 1, \dots, N^{obj}, && \text{(2.65f)}
\end{aligned}$$

where $d^{ref}_{\Pi_i\Pi_j}$ and $\Delta e^{ref}_{y,\Pi_i\Pi_j}$ are desired reference distances between vehicles Π_i and Π_j, and ϵ_d, $\epsilon_{\Delta e_y}$ and ϵ_v are positive and small. A typical choice may be a constant $d^{ref} > 0$ such that $d^{ref}_{\Pi_i\Pi_j} = d^{ref}$ and $\Delta e^{ref}_{y,\Pi_i\Pi_j} = 0$.

Property 2. *Let $d_{\Pi_i\Pi_j}(t) > d^{min}_{\Pi_i\Pi_j}(t)$, for $\Pi \in \mathbb{Z}_{++}^{N^{obj}}$ according to Definition 1 with $\Pi_i \neq \Pi_j,\ j = i+1,\ \forall i, j = 1, \dots, N^{obj}$ and $\forall t$ such that $s_{\Pi_i}(t) > s_{\Pi_j}(t)$, where $d^{min}_{\Pi_i\Pi_j}(t)$ denotes the minimal distance that ensures collision-free braking is always possible when operating all vehicles autonomously. Then, selecting Π according to descending s-coordinates guarantees collision-free safety when operating all vehicles coordinatedly.*

Proof. The set of operating multiple automated vehicles by controlling each one of them autonomously is included in the set of operating multiple automated vehicles coordinatedly, which proves the property. □

A relaxation of Property 2 is to additionally allow for OA-maneuvers to maintain safety. An implications of Property 2 is that coordination of automated vehicles allows to lower longitudinal safety distances $d^{\min}_{\Pi_i\Pi_j}(t)$. The minimal communication delay between two vehicles Π_i and Π_j is at least T_s. Suppose vehicle trajectories for non-braking and braking operation are known such that

$$s_{\Pi_i}(t) = \begin{cases} f_{\Pi_i}(t), & \forall t < t^b, \\ f^b_{\Pi_i}(t - t^b), & \forall t \geq t^b, \\ s_{\Pi_i}(\bar{t}_{\Pi_i}), & \forall t \geq \bar{t}_{\Pi_i} \geq t^b, \end{cases}$$

$$s_{\Pi_j}(t) = \begin{cases} f_{\Pi_j}(t), & \forall t < t^b + pT_s, \\ f^b_{\Pi_j}(t - t^b - pT_s), & \forall t \geq t^b + pT_s, \\ s_{\Pi_j}(\bar{t}_{\Pi_j}), & \forall t \geq \bar{t}_{\Pi_j} \geq t^b + pT_s, \end{cases}$$

for $p \in \mathbb{Z}_{++}$, and where t^b denotes the time when vehicle Π_i initializes braking, and $\bar{t}_{\Pi_i}$ and $\bar{t}_{\Pi_j}$ indicating the times when reaching standstills by vehicles Π_i and Π_j, respectively. Then, given vehicle states at time t and a communication delay pT_s, we can determine a *lower bound* on $d^{\min}_{\Pi_i\Pi_j}(t)$ from $d_{\Pi_i\Pi_j}(t) = s_{\Pi_i}(t) - s_{\Pi_j}(t) > d^{\min}_{\Pi_i\Pi_j}(t)$, $\forall t$. This is relevant for the realization of any platooning objectives.

The control system for an automated vehicle under Assumption 1 is capable of, in general, suboptimally (since a decoupled method is employed) realizing both Objectives 1 and 2. The distinction is made on the switching-rule level. For throughput maximization, we constantly encourage OA-maneuvers, as long as there exists a free neighbor-lane gap permitting these. For the platooning application, we enforce the ACC-state. In contrast, to conservative autonomous driving, we here allow vehicles to temporarily accelerate. Once the platooning state is reached, we additionally coordinate vehicles to follow varying e_y-trajectories of the leading vehicle (e_y-reference tracking). In both cases, the cooperative nature allows for small safety margins.

Remark 5. *A desired ordering of vehicles $i = 1, \ldots, N^{obj}$ within a platoon can be achieved by the selection of reference velocities v_i^{ref}, $\forall i = 1, \ldots, N^{obj}$, and the switching between one of the Objectives 1 and 2 over subsequent sampling times. This is relevant for formation driving, not only on-road but also off-road along virtual reference centerlines.*

Algorithm 6: One-directional traffic flow, @every T_s **do**:

1 Within a road segment of interest including N^{obj} automated vehicles, select temporary priority scheme Π' according to descending s-coordinates.
2 **for** *each vehicle* $i \in \Pi'$ **do**
3 **if** *there exists a deviation of priority of vehicle i from Π' and Π (from the last sampling time) with another automated vehicle j* **then**
4 **if** $|s_i - s_j| > s_{OA}^{safety}$ **then**
5 Swap priorities of i and j within Π.
6 Tuning 1: select v_i^{ref}, $\forall i = 1, \ldots, N^{\text{obj}}$.
7 Tuning 2: select one of the two objectives:
- throughput maximization,
- quickly reaching of a platooning state.
8 **for** *each vehicle* Π_i, $i = 1, \ldots, N^{obj}$ **do**
9 Take the already computed reference trajectories of vehicles $\Pi_1, \ldots, \Pi_{i-1}$ in the following computations [155] into account:
- velocity-adjusted object mapping to the corridor,
- driving mode selection,
- corridor planning using the geometric path planner.
10 Obtain reference trajectories, constraints and weighting matrices for vehicle Π_i such that processable by the controller onboard the vehicle.

Algorithm 6 summarizes the findings and is executed as Step 3 of Algorithm 5 for one-directional traffic flow control. Value $s_{\text{OA}}^{\text{safety}}$ (e.g., 30m) represents an arbitrary safety parameter to change priorities only after full completion of an OA-maneuver. Notice that for the one-directional traffic flow case, we allow a *nonlane road* to maximally exploit varying vehicle dimensions and agility capabilities.

Ultimately, note that for the realization of Objectives 1 and 2, in a

wireless VANET there is not necessarily a need for a coordinating entity. The priority scheme according Algorithm 6 can be assigned in a fully distributed manner. Important is solely the passing of information within the network. Since by Assumption 1 every vehicle Π_i, $\forall i = 1, \dots, N^{\text{obj}}$, can also act autonomously, it can check its priority by awareness of the other $N^{\text{obj}} - 1$ automated vehicles and individually compute its corridor path considering the reference trajectories of prioritized vehicles $\Pi_1, \dots, \Pi_{i-1}$. However, if we additionally seek to reach a particular ordering of vehicles according to Remark 5, a coordinating entity becomes necessary for the assignment of reference velocities and switching between throughput maximization and platooning objectives.

Bi-directional Traffic Flow

We refer to a tube-like road as *bi-directional* if its traversal is admissible with heading direction towards positive as well as negative s. Let the two traversal directions be denoted by $h = 1$ (facing $s > 0$) and $h = 2$ (facing $s < 0$), respectively. Let the heading direction of vehicles $i = 1, \dots, N^{\text{obj}}$ be denoted by $h_i(t)$. We assume that vehicles maintain their traversal direction, i.e., $h_i(t) = h_i(0) = h_i,\ 0 \leq t \leq T$, where T denotes the time horizon, and define the velocity sign as $v_i > 0$ if $h_i = 1$ and $v_i < 0$ for $h = 2$. Let the set of vehicles within the road segment of interest be denoted by $\mathcal{N}^{\text{obj}} = \{1, \dots, N^{\text{obj}}\}$. We further define $\mathcal{N}^{\text{obj},1} = \{i \in \mathcal{N} : h_i = 1\}$ and $\mathcal{N}^{\text{obj},2} = \{i \in \mathcal{N} : h_i = 2\}$. Let priority schemes Π^1 and Π^2 correspond to $i \in \mathcal{N}^{\text{obj},1}$ and $i \in \mathcal{N}^{\text{obj},2}$. Let a *bi-directional traffic flow conflict* be defined as a vehicle constellation in which a head-to-head collision between at least two vehicles $i \in \mathcal{N}^{\text{obj},1}$ and $j \in \mathcal{N}^{\text{obj},2}$ becomes unavoidable if not conducting an OA-maneuver for at least one of i and j. We initialize $t = 0$ at the time of detection of a bi-directional traffic flow conflict. Let the centerline of *lanes* $m \in \mathcal{N}^{\text{lanes}} = \{1, \dots, N^{\text{lanes}}\}$ be denoted by $e_y^m(s) \in [-0.5e_y^w(s), 0.5e_y^w(s)]$. For uniform lane-widths we have $e_y^m(s) = -0.5e_y^w(s) + (0.5 + m - 1)\frac{e_y^w(s)}{N^{\text{lanes}}}$. For the resolution of a bi-directional traffic flow conflict we assume space-invariant centerline levels and thus write $e_y^m(s) = e_y^m$.

Remark 6. *Let there be N^{obj} automated vehicles and N^{lanes} lanes of uniform width and $|v_i(t)| > 0,\ \forall t,\ \forall i \in \mathcal{N}^{obj}$ within a particular road segment. Suppose Π^1 and Π^2 are determined separately for $i \in \mathcal{N}^{obj,1}$ and $i \in \mathcal{N}^{obj,2}$, respectively. Then, collision-free bi-directional traffic flow can, in general, not be guaranteed by solely concatenating Π^1 and Π^2 as $\Pi = [\Pi^1,\ \Pi^2]$ or $\Pi = [\Pi^2,\ \Pi^1]$. This can easily be seen from a counterexample. Suppose for $N^{lanes} \leq N^{obj} - 1$ there are vehicles $i = 1, \dots, N^{lanes}$ with $h_i = 1,\ s_i(t) = s(t),\ e_{y,i}(t) = e_y^m,\ m = 1, \dots, N^{lanes},\ \forall t$ and uniform $v_i(t) = v$, such that because of the non-zero velocity operation assumption a vehicle $j \in \mathcal{N}^{obj,2}$ cannot collision-free pass unless at least one of the vehicles $i = 1, \dots, N^{lanes}$ makes space by performing a braking- and/or OA-maneuver. Then, by Definition 1, setting $\Pi = [\Pi^1,\ \Pi^2]$ guarantess a head-to-head collision.*

Remark 6 implies that for the resolution of a bi-directional traffic flow conflict, a collision-free coordination can, in general, only be achieved by either *threading* of vehicles from both directions, or, by the *assignment of lanes* to each heading direction. For a threading-approach in the context of quadrocopters using a sequential convex programming, see [14].

In contrast, we focus on the second solution approach and assume there are N^{lanes} of uniform width, $\frac{e_y^w(s)}{N^{\text{lanes}}}$, that can be traversed by any vehicle $i = 1, \dots, N^{\text{obj}}$. For the assignment of each lane to exactly one of two heading directions, we formulate the following binary integer linear program (BILP):

$$\min_{u_{mn}^h} \sum_{h=1}^{2} \sum_{m=1}^{N^{\text{lanes}}} \sum_{n=1}^{N^{\text{lanes}}} c_{mn}^h u_{mn}^h \tag{2.66a}$$

$$\text{s.t.} \quad \sum_{n=1}^{N^{\text{lanes}}} u_{mn}^h = 1,\ \forall m = 1, \dots, N^{\text{lanes}},\ \forall h = 1, 2, \tag{2.66b}$$

$$u_{mn}^h + u_{mn}^{\tilde{h}} = 1,\ \forall m, n = 1, \dots, N^{\text{lanes}}, \quad \forall h \in \{1, 2\}, \tilde{h} \in \{1, 2\} \backslash h, \tag{2.66c}$$

$$u_{mn}^h \in \{0, 1\},\ \forall m, n = 1, \dots, N^{\text{lanes}},\ \forall h \in \{1, 2\}, \tag{2.66d}$$

where $m, n \in \{1, \dots, N^{\text{lanes}}\}$ denote a lane-number. Cost coefficients c_{mn}^h are discussed below. Integer variable $u_{mn}^h = 1$ signals automated vehicles currently driving on lane m in direction h are assigned to change

lane to lane number n. Correspondingly, $u^h_{mn} = 0$ prohibits the same lane change. Constraint (2.66b) indicates that all cars currently on lane m with heading h will transfer to exactly *one* other lane $n \in \{1, \dots, N^{\text{lanes}}\}$. An alternative would be the admittance of each object individually transfering to any other lane (threading). By (2.66b) there is at least one lane assigned to each traversal direction. Constraints (2.66b) are further motivated by the cooperative problem nature; the lane transition can be conducted in parallel with all vehicles commencing the lane change simultaneously. Constraints (2.66c) are to assign every lane to exactly one of the two traversal directions. Constraints (2.66d) ensure optimization variables u^h_{mn} to be binary.

Proposition 6. *The solution of the LP-relaxation of BILP* (2.66)*, formulated for the assignment of any number of lanes, $N^{lanes} \in \mathbb{Z}_{++}$, to exactly one of two heading directions, $h \in \{1, 2\}$, is integer feasible, and thus solves* (2.66) *as well.*

Proof. We can easily summarize (2.66) as $\min\{c^\mathsf{T} x : Ax = \mathbb{1},\ x_l \in \{0, 1\}, \forall l = 1, \dots, 2N^{\text{lanes}} N^{\text{lanes}}\}$. Its LP-relaxation reads $\min\{c^\mathsf{T} x : Ax = \mathbb{1},\ x \geq 0\}$. By [332], if $\tilde{A}$ is *totally unimodular*, the LP $\min\{\tilde{c}^\mathsf{T}\tilde{x} \ : \tilde{A}\tilde{x} = \tilde{b},\ \tilde{x} \in \mathbb{R}^n_+\}$ has an integral optimal solution for all integer vectors $\tilde{b}$ for which it has a finite optimal value. It thus remains to show that A associated with the LP-relaxation of (2.66) is totally unimodular. By [180], a matrix A is totally unimodular if: (i) each entry is 0, 1 or -1; (ii) each column contains at most two non-zeros; (iii) the set $\mathcal{N}$ or row indices of A can be partitioned into $\mathcal{N}_1 \cup \mathcal{N}_2$ such that in each column l with two non-zeros we have $\sum_{m_1 \in \mathcal{N}_1} a_{m_1 l} = \sum_{m_2 \in \mathcal{N}_2} a_{m_2 l}$. Condition (i) is trivially true from (2.66b) and (2.66c). Regarding (ii), for every column $l = (\overline{m} - 1)N^{\text{lanes}} + \overline{n} + (\overline{h} - 1)N^{\text{lanes}} N^{\text{lanes}},\ \overline{m}, \overline{n} \in \mathcal{N}^{\text{lanes}},\ \overline{h} \in \{1, 2\}$, there is $\sum_{n=1}^{N^{\text{lanes}}} u^{\overline{h}}_{\overline{m}n} = 1$ and $u^{\overline{h}}_{\overline{m}\overline{n}} + u^{\bar{\overline{h}}}_{\overline{m}\overline{n}} = 1$, which implies that per column of A there are exactly two nonzero coefficients, which are here equal to 1. For (iii), we partition as $\mathcal{N}_1 = \{1, \dots, 2N^{\text{lanes}}\}$ and $\mathcal{N}_2 = \{2N^{\text{lanes}} + 1, \dots, 2N^{\text{lanes}} + N^{\text{lanes}} N^{\text{lanes}}\}$. Then $\sum_{m_1 \in \mathcal{N}_1} a_{m_1 l} = 1$ by (2.66b) and $\sum_{m_2 \in \mathcal{N}_2} a_{m_2 l} = 1$ by (2.66c) using the previous argument that per column l there are exactly two nonzero coefficients, both equal to 1. This concludes the proof. □

By Proposition 6 it is thus possible to solve (2.66) efficiently as a LP with $2N^{\text{lanes}} N^{\text{lanes}}$ variables and $2N^{\text{lanes}} + N^{\text{lanes}} N^{\text{lanes}}$ equality constraints,

which makes the approach suitable for real-time implementation. Remaining is the discussion of c^h_{mn} in (2.66a), the cost for transitioning from lane m to n with direction h. Denoting the number of automated vehicles on lane m facing direction h by N^h_m, one option for the cost cefficients is $c^h_{mn} = N^h_m \cdot |n-m|,\ \forall m, n = 1, \dots, N^{\text{lanes}},\ \forall h = 1, 2$, which assigns a cost 1 per object requiring a lane change and additionally employs the term $|n-m|$ to penalize multiple lane-skipping changes. A second option is motivated as follows. We define $s^{m,h}_{i^\star} = \{s_{i^\star}(0) : i \in \mathcal{N}^{\text{obj},h},\ s_{i^\star}(0) = \max_i\{s_i(0)\}$ if $h = 1$ and $s_{i^\star}(0) = \min_i\{s_i(0)\}$ if $h = 2,\ |e_{y,i}(0) - e^m_y| < \epsilon\}$ with $\epsilon > 0$ and small, and the corresponding velocity $v^{m,h}_{i^\star} = v_{i^\star}(0)$. From setting $s^{m,h}_{i^\star} + \Delta t^h_{mn} v^{m,h}_{i^\star} = s^{n,\tilde{h}}_{j^\star} + \Delta t^h_{mn} v^{n,\tilde{h}}_{j^\star}$, we compute $\Delta t^h_{mn} = \frac{s^{n,\tilde{h}}_{j^\star} - s^{m,h}_{i^\star}}{v^{m,h}_{i^\star} - v^{n,\tilde{h}}_{j^\star}}$ and set arbitrarily a high $\Delta t^h_{mn} = 100$ if there does not exist any vehicle on one or both of lanes m and n. We do not set $\Delta t^h_{mn} = \infty$ to still distinguish between multi-lane skipping. Thus, a second option is

$$c^h_{mn} = \frac{1}{\min_{\tilde{n}} \{\Delta t^h_{m\tilde{n}}\}} |n-m|, \tag{2.67}$$

with $\tilde{n} \in \{m+1, \dots, n\}$ if $n > m$, and $\tilde{n} \in \{m-1, \dots, n\}$ if $n < m$. The interpretation is that we penalize the inverse approximate time until a crash frontal between the first car of originally lane m, facing direction h and now transfering to lane n, with the time-closest vehicle facing the counterdirection $\tilde{h}$ on *any* of the lanes $\tilde{n}$ between m and n. The time is approximate since we assume an immediate lane change not modeling actual transient times for the lane change and variations in speed throughout the lane change. Combining the two options, we define

$$c^h_{mn} = \left(N^h_m + \mu \frac{1}{\min_{\tilde{n}} \{\Delta t^h_{m\tilde{n}}\}} \right) \cdot |n-m|, \tag{2.68}$$

with trade-off parameter μ. Small μ encourage only very few objects to change their lane but may situation-dependent invoke dangerous multi-lane skipping. In contrast, a high μ is more safety-oriented but may require lane changes by a majority of vehicles. Algorithm 7 summarizes

Algorithm 7: Bi-directional traffic flow, @every T_s **do**:

Order the automated vehicles according traversal direction into two groups, $\mathcal{N}^{\text{obj},1}$ and $\mathcal{N}^{\text{obj},2}$.
For each direction $h \in \{1, 2\}$, conduct Steps 1 to 5 of Algorithm 6 to obtain Π^h, whereby for $h = 1$ and $h = 2$ the sorting is conducted according to descending and ascending s-coordinates, respectively.
Select c^h_{mn} according (2.67).
Solve the LP-relaxation of (2.66) for the assignment of N^{lanes} to exactly one of two traveling directions, $h \in \{1, 2\}$.
for *each* $h = 1, 2$ **do**
    for $m = 1, \ldots, N^{lanes}$ **do**
        find $n^\star = \{n \in \mathcal{N}^{\text{lanes}} : u^h_{mn} = 1\}$.
        for *all* $\{i \in \mathcal{N}^{obj,h} : |e_{y,i}(0) - e^m_y| < \epsilon\}$ **do**
            $e^{\text{ref}}_{y,i} = e^{n^\star}_y$.
Concatenate the updated Π^1 and Π^2 to Π.
Conduct Steps 6 to 10 of Algorithm 6 using Π, whereby using the updated references on $e^{\text{ref}}_{y,i}$, $\forall i = 1, \ldots, N^{\text{obj}}$.
Store the solution as the *current* solution.
if `maxIter` > 1 **then**
    initialize $\mu = 0$.
    for `iter` $= 1, \ldots,$ `maxIter` **do**
        Select c^h_{mn} according (2.68) and conduct Steps 4:11.
        Check resulting trajectories for feasibility:
        If they are feasible, update this solution as the *current* solution and exit. Otherwise, increase μ.
Return the *current* solution.

Table 4: Average computation times $\bar{\tau}$ in milliseconds. For bi-directional traffic flow, the computation times for solving the LP-relaxation of (2.66) via MATLAB's `linprog` and, alternatively, for solving the BILP (2.66) directly by enumeration are denoted by $\bar{\tau}_{\texttt{linprog}}$ and $\bar{\tau}_{\text{lp,enum}}$, respectively. For comparison, we state $\bar{\tau}_{\text{corridor}}$ for the solution of the corridor planning problem. Regarding the LSV-MPC problem, $\bar{\tau}_{\text{qp,build}}$ includes linearization, discretization and building of the QPs. The average computation times using MATLAB's `quadprog` for the solution of the QPs are denoted by $\bar{\tau}_{\texttt{quadprog}}$. The platooning and throughput maximization objectives are abbreviated by (P) and (TM), respectively. The average velocity-dependent spatial-based prediction horizon is $\bar{N}$. Times $\bar{\tau}_{\text{corridor}}$, $\bar{\tau}_{\text{qp,build}}$, $\bar{\tau}_{\text{quadprog}}$ and $\bar{N}$ are for each experiment further averaged over all five automated vehicles.

	Figure 39 (P)	Figure 39 (TM)	Figure 40
$\bar{\tau}_{\texttt{linprog}}$	-	-	6.1
$\bar{\tau}_{\text{lp,enum}}$	-	-	0.1
$\bar{\tau}_{\text{corridor}}$	0.9	0.6	1.3
$\bar{\tau}_{\text{qp,build}}$	17.1	13.7	15.5
$\bar{\tau}_{\texttt{quadprog}}$	78.9	41.4	48.5
N	38	33	36

the findings and is executed as Step 3 of Algorithm 5 for bi-directional traffic flow control. Steps 13 until 18 are optional to better account for the aforementioned trade-off. The design parameter $\texttt{maxIter} \in \mathbb{Z}_+$ denotes the maximum number of μ-iterations. A simple feasibility check is $\sqrt{d_{ij}(t)^2 + \Delta e_{y,ij}(t)^2} > l^{\min}$, $\forall i, j \in \mathcal{N}^{\text{obj}}$, $i \neq j$, $\forall t$, ensuring a minimum safety distance $l^{\min}$ between vehicles. Step 10 of Algorithm 7 makes the assumption that all lane changes, as assigned by the solution of (2.66), will be completed *before* the crossing of vehicles along the road centerline coordinate. This makes the exact merging technique less of an issue and our preferred method is thus concatenation.

Finally, we point out the suitability of the BILP-formulation in combination with our controller described in [155]. By the assignment of reference set points on state e_y corresponding to any of the respective lane centerlines, the optimized lane assignment from (2.66) can easily be realized. This is because of the control design implemented entirely

spatial-based in a road-aligned coordinate system. In general, for the realization of the lane changes multiple driving mode activations such as braking, OA and ACC are required.

The solution of the LP-relaxation of (2.66) can be computed *either* locally by one of the vehicles (car with most computational power acting as service) within the VANET, *or* by an independent entity, e.g., a web-based coordination service.

2.3.3 Numerical Simulations

For all three numerical simulations we assumed a challenging inter-vehicle communication range of only 100m. All automated vehicles are described by a nonlinear dynamic bicycle model with throttle, brakepedal position and steering as control inputs, see [155]. For the control of *each* automated vehicle, we use the framework from the aforementioned reference. The *coordination* of multiple vehicles then follows Algorithms 5, 6 and 7. Computation times are summarized in Table 4. For comparison, we also state times for the solution of the corridor planning and the LSV-MPC problem. All simulations are conducted on a laptop running Ubuntu 14.04 equipped with an Intel Core i7 CPU @2.80GHz×8, 15.6GB of memory, and using MATLAB 8.6 (R2015b). For visualization of the dynamics, animated simulations of the experiments are available at `http://dysco.imtlucca.it/mogens/sim_coordinated_driving.htm`.

One-directional Traffic Flow

Figure 39 illustrates the results of a one-directional traffic flow simulation. We compare two scenarios. First, given the starting states of all vehicles as indicated in Figure 39 we aim at quickly reaching a platoon adapted in velocity to the vehicle most advanced along the road centerline. In the second scenario, we seek throughput maximization by allowing all vehicles to travel at their reference velocities and therefore encourage overtaking in case of available free neighbor-lane gaps.

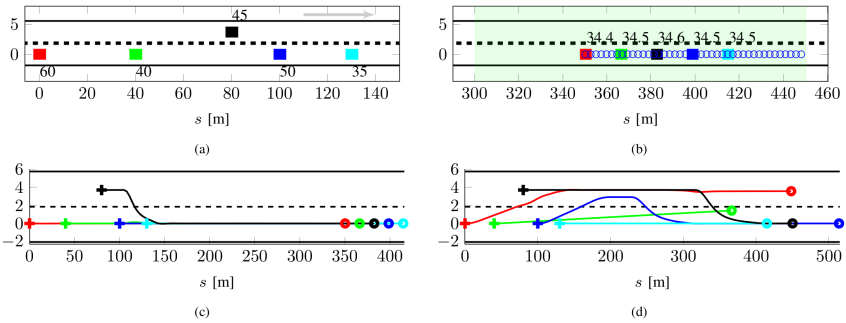


Figure 39: (a) Starting positions of five automated vehicles. The numbers close to the vehicle indicate their initial velocity in km/h. The color is an identifier of each vehicle. (b) Platoon driving, or coordinated adaptive cruise control–CAAC, from the perspective of the red vehicle at the last time step. The light green area indicates the range field centered around the red vehicle. The blue circles indicate the predicted reference trajectory of the red vehicle. (c) Trajectories of all five automated vehicles for the reaching of a platoon formation. The crosses indicate the starting and the circles the ending positions after a simulation time of 30s. (d) Trajectories of all five automated vehicles when the global objective is throughput maximization. The final position coordinates of all cars are naturally more advanced along the road centerline in comparison to the platoon driving scenario.

Bi-directional Traffic Flow

The results of an experiment for a bi-directional traffic flow conflict are visualized in Figure 40. Without a vehicle coordination, a head-to-head collision will become unavoidable on both of the two available lanes. Table 4 indicates that, for the given two-lane case, solving (2.66) directly by enumeration is more efficient than solving its LP-relaxation. Ultimately, for cooperative driving in case of bi-directional traffic flow, a mapping between two road-aligned coordinate systems for both heading directions is required.

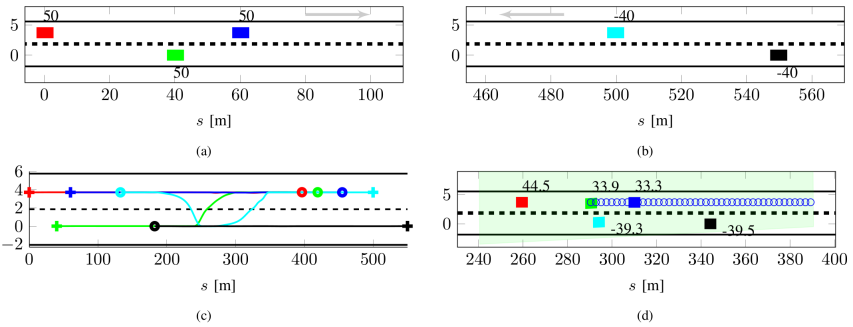


Figure 40: (a) Starting position of three automated vehicles heading as indicated by the arrow. The numbers close to the vehicle indicate their initial velocity in km/h. (b) Starting positions of two other automated vehicles heading in the opposite direction, as signaled by the arrow. (c) Trajectories of all of the five automated vehicles for the solution of the given bi-directional traffic flow conflict. The crosses indicate the starting positions and the circles denote the ending position after a simulation time of 30s. (d) Display of the time instance shortly after all five vehicles have performed the lane change as assigned by the coordination service. The blue circles indicate the predicted reference trajectory of the green vehicle.

2.3.4 Hierarchical Controller Parametrization

The hierarchical controller parametrization for multi-vehicle motion planning is summarized by Figure 38 and Algorithm 5.

2.4 Discussion and Conclusion of Chapter

Summary

In this chapter, it was distinguished between two major topics: single- and multi-vehicle motion planning. For the former topic, it was further differentiated between motion planning by a) MPC and b) neural networks, respectively. For multi-vehicle motion planning, a method for single-vehicle motion planning is a prerequisite, before a multi-vehicle coordination layer is added.

For the MPC approach, findings can be summarized as follows.

1. For single-vehicle motion planning by MPC, a *spatial* system parametrization is preferred over a more standard time parametrization. It was discussed how a spatial parametrization is ideal for navigation in static and quasi-static obstacle settings. In these scenarios, a time-parameterized LTV-MPC formulation may already easily fail, in particular, when *reference velocities* are not selected carefully.

2. For general motion planning, a LSV-MPC formulation is required to sufficiently capture system nonlinearities. A space-*invariant* system model is not sufficient. This has important implications. First, theory for linear MPC can only limitedly be applied. Second, *online* linearization and discretization become necessary. Then, especially for higher dimensional models (such as dynamic instead of kinematic vehicle models), the corresponding online computational burden to "build" QPs or LPs is also not negligible, and adds to the computiational burden to solve QPs or LPs online.

3. With regard of the fundametal MPC-formulation setup, the *LP*-based approach in [148] is clearly preferable over the QP-based approach of [155]. This is for two main reasons. First, the min-max objective in the LP-formulation in combination with driving corridor constraints naturally circumvents the need for objective function tuning weights, which are characteristic and influential for a more standard QP-based reference tracking MPC formulation.

Thus, for the LP-minmax approach, references are only needed for linearization and discretization, but not for a reference tracking objective. Second, a minmax objective on absolute steering angles and differences thereof naturally increases safety. This is since minimized steering actuation implies lower path curvature, and thereby naturally also a higher admissible vehicle velocity that still permits vehicle operation within tire friction limits.

4. A *sequential* LP (SLP) approach is preferred. This is for two reasons. For the spatial prediction horizon and as a function of path curvature, the solution of the first LSV-MPC problem can be used to compute maximum admissible velocities permitting operation within vehicle tire friction limits (assuming a friction circle model with coefficient μ). Then, these velocity bounds can be added as maximum velocity constraints to the second LSV-MPC problem for velocity control refinement. The second reason is that for maneuvering in tight spaces, where large steering actuation is required, transition dynamics and vehicle dimension constraints are strongly dependent on underlying references used for linearization and discretization. Here, sequential iterations can help to refine solutions.

The neural network approach can be summarized as follows. First, within the context of automated vehicles, a method for the design of model-based feedforward controllers parameterized by deep neural networks was presented. Second, for this method, a suitable closed-loop architecture was identified, and a simple gradient-free reinforcement learning algorithm labeled TSHC was developed for the identification of network parameters. The concept of a) training on separate tasks with the purpose of encoding motion primitives within the network, and b) employing sparse rewards in combinations with virtual actuator constraints on velocity in setpoint proximity were specifically advocated. Third, the presented method is not limited to automated driving. Most real-world learning applications for control systems, especially in robotics, are characterized by a) sparse rewards, and b) the availability of high-fidelity system models that can be leveraged for offline training.

More remarks on motion planning by MPC are made.

1. As stated at the beginning of Section 2.1, only QPs and LPs were considered as candidate parametrizations for MPC. In this thesis, QP- or LP-*solvers* were not discussed. This would be subject of next work. Since LPs are preferred for motion planning by MPC, natural starting points are a) the *Simplex Method* [290] and b) warmstarting.

2. While MPC is famous for its ability to incorporate system constraints in a systematic way, it has several disadvantages. These include that linearization and discretization must be conducted *online* and importantly space-varyingly over the prediction horizon. As illustrated, this results in a) non-negligible computational burden, and b) in an approximation of the original nonlinear dynamics. While the linearization step is a linear mapping for each sampling space, the discretization step is a linear mapping only for a kinematic bicycle model. In contrast, for dynamic vehicle models it requires the evaluation of matrix exponentials (for exact discretization). Other disadvantages are the need to select hyperparameters, in particular, to design the relation between system sampling time T_s, prediction horizon N, and sampling space D_s, as well as state constraints (in particular, on e_y). These design steps also influence computational burden and the degree of approximations of the original nonlinear system. In short, MPC design requires a significant amount of trade-offs and approximations.

3. While a time-parameterized LTV-MPC formulation is problematic because of its dependence on reference velocities, the preferred space-parameterized LSV-MPC formulation is dependent on the reference path. This is since all states and controls are expressed *uniquely* along spatial coordinate s along the reference path (the road centerline). Thus, for general motion planning with significant steering, *sequential programming* (SP) iterations will always be required (since there cannot exist, e.g., two $e_{y,j}$ for the same s_j). This holds for any NMPC solution method. Thus, even when solving the motion planning problem by another *nonlinear programming*

method (i.e., a method different from SLP), sequential iterations will still be required to iterate over the reference path.

4. For simple road centerline tracking applications (e.g., on highways or country roads), aforementioned sequential programming approach is not necessarily required. However, for tight maneuvering it is. In this perspective, different control methods could be employed for different velocity ranges. This, however, is in general undesirable, since it requires to design switching rules, and thereby adds to aforementioned trade-offs and approximations for MPC.

5. A *real-time* optimization-approach such as MPC (or also RRTs [242]) is most suitable when planning over a large spatial horizon and for scenarios that are difficult to pre-train offline. An example is visualized in Figure 41, where anticipative steering accounts for the *entire* and given scenario-specific multi-obstacle configuration space. Another example is trajectory planning over *multiple* curves. However, computational complexity increases with increasing optimization horizons. In addition, a *dynamic environment* with dynamic obstacles requires frequent resetting of *setpoints* in vehicle-proximity and in general does not permit to plan over a large horizon (e.g., over multiple curves or obstacles). Therefore, in dynamic environments, the neural approach with encoded motion primitives may be more suitable and also safer when a) trained offline on a high-fidelity vehicle model, and b) employed online in combination with a suitable recursive setpoint selector.

Future Work

The last comment may be the starting point of future work based on the TSHC-algorithm from [142]. Large-scale simulation experiments are sought. Therefore, the next steps include a) vehicle model selection, b) training task setup with motion primitives selection, and c) neural network parametrization. Furthermore, it may be considered to extend the

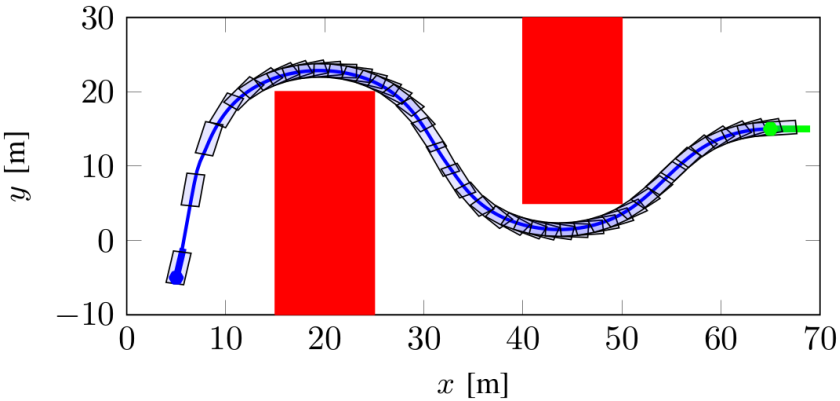


Figure 41: Example from [156], when motion planning over a large spatial horizon and for the entire (static) multi-obstacle configuration space. According to the optimization problem formulation, obstacles are avoided tightly. In practice, small safety margins may be added to obstacle contours.

state space with time t to permit time scheduling along waypoints similar to the method from [148].

Recursive setpoint selections must be conducted in real-time. Therefore, an algorithm that recursively updates setpoints as a result of collision checks, obstacle corners, road centerline, obstacle free areas and the like must be developed. An important testbed are parking applications, which additionally involve the switching between forward and backward motion. A future setpoint selection algorithm may be founded on the graph-based geometric corridor planner from Section 2.1.14, which is fast, can handle non-convex problems, offers an appropriate (for minimized steering) accumulated least-heading varying optimization criterion, and permits to derive worst-case search complexities.

Future work on multi-vehicle motion coordination may address the question of how to adapt priorities to quickly reach a specific desired ordering or formation of a vehicle group. Formations may also include lateral displacements between vehicles. The basic framework from [154] using decoupled prioritized path planning can be maintained. Each automated vehicle may also be controlled by a neural network controller.

Chapter 3

Vehicle Routing

Agriculture is considerd as a logistics case study for vehicle routing. Developed concepts are, with some modifications, transferable to, for example, public transport routing or for robot taxi dispatch.

3.1 Path Planning for Area Coverage

This section summarizes [146]:

- M. Graf Plessen, "Path planning for area coverage," *WO Patent App. PCT/EP2016/072966*, June 8 2017.

A general framework and three specific algorithms for plath planning for *area coverage* are developed.

3.1.1 Introduction

For the coverage of an agricultural field (or work area in general), paths for various vehicles of different working widths and scope have to be devised. By adequate selection of field coverage paths, efficiency with respect to time, energy and utility of available area can be increased. Time is important, in particular, during harvest due to changing weather conditions and the subsequent availability of only a limited number of harvesting hours. Energy savings (usually Diesel) and likewise the reduc-

tion of operating hours and wear of the vehicle are of interest not only for cost reduction, but also for environmental reasons and regulatory exhaust emission standards. Furthermore, by appropriate selection of field coverage paths, the area repressed by the tractor lanes can be minimized and thus reducing crop damage. This also implies the maximization of area available for planting and subsequently monetary gain.

Throughout a work-cycle, a field may be ploughed, sowed or planted, fertilized, sprayed and harvested. In general, operating widths vary. However, some operations are conducted multiple times in a repetitive manner and using the same operating width such as, e.g., 36m. This implies the possibility of using a particular created path plan repeatedly.

In nature, field contours are often irregularly shaped and include non-plantable islands prohibited from trespassing (trees, power line poles or masts and the like). Especially then, it is not obvious how to efficiently plan paths for complete field coverage. Working sequentially lane-by-lane has the advantage of being intuitive, simple to implement and enabling to visually accurately detect and control the area already worked, for example, during ploughing, and thus avoiding overlapping. More efficient field coverage and path planning techniques can be developed, that do not necessarily require sequential lane-by-lane operations. By the advent of modern sensor and actuation technologies, these alternative more efficient field coverage path plans can be realized, and may even be executed by an autonomous agricultural machinery.

3.1.2 Algorithms

Graph Theoretical Background

The presented methods make use of graph theory. For background, see [64]. A graph is composed of a set of vertices (also called nodes) and a set of edges. With every edge a nonnegative weight is associated. Here, this weight may be represented by the path length of the edge. Graphs are used to model transitions between particular location positions, also called nodes, along predefined route segments, also called edges, and here also referred to as lane-segments or simply lanes. Nodes are here

also referred to as cross-points and defined by connecting edges. For example, we define lane-segment a-b as an edge or a direct path connecting two cross-points or nodes a and b. The term transition graph is therefore used. An *undirected graph* (in contrast to a *directed graph*) is composed of bidirectional edges, i.e., of edges that may be traversed from either side. A graph is connected if it contains a path between any pair of vertices, where a path is defined as a sequence of edges, or, in other words, a sequence of pairwise adjacent vertices whose edges connect two vertices. An undirected connected transition graph is here denoted by T. The degree of a vertex v in T is the number of edges of T incident with v. For illustration, see node F6 in Figure 42 which has degree 3, or, more generally speaking, odd-degree since there is an odd number of edges (here 3) incident. In contrast, the start-node 0, see F8, is even-degree with 2 edges incident. A *tour* of T is a closed walk that traverses each edge of T at least once. A graph is *Eulerian* if there exists a closed traversal using all edges exactly once. If a graph is undirected, necessary and sufficient conditions for a graph being Eulerian are that the graph must be connected and all vertices must be even degree, see [82]. A connected undirected graph T can be made Eulerian by the augmentation of T, i.e., by replicating (also called duplicating) some edges of T until the resulting T^* is an Eulerian graph. A connected undirected graph T always has an even number of odd-degree vertices, see [101]. Therefore, for a connected undirected graph T the augmentation can be accomplished by solving a *matching problem*, i.e., duplicating edges (and their corresponding weights) to link odd-degree vertices. Usually, but not necessarily, the augmentation is conducted in a least-cost manner, i.e., finding odd-degree vertex-pairings such that the sum of the weights of the corresponding connecting edges is minimal, see [82]. Assuming an Eulerian graph T^*, an Eulerian cycle (also called Eulerian tour) describes a tour which traverses each edge of T^* exactly once. An Eulerian cycle in an undirected Eulerian graph T^* can be determined by, for example, the *End-Pairing algorithm* [101]. The total path length of the Eulerian cycle is then the sum of all edges of T^*. In the following, for brevity, it is referred to closed areas around trees, stones, pools, power line poles, masts or similar areas and obstacles not

suited for planting and prohibited from trespassing by agricultural machines as *tree-islands* or *islands*.

Within this description, the term *180°-turn* is repeatedly occurring. Thereby is meant a cross-point sequence of a-b-a, where a and b are neighboring or connected cross-points according to the underlying transition graph. It is here emphasized that the term 180°-turn explicitly refers to the direction of the vehicle or machinery movement. Thus, it can, but does not necessarily have to, imply a turning maneuver of the vehicle, but likewise may suggest a back-forth movement, i.e., driving forward along edge a-b and reverse on b-a.

Here, "online" means that results of the method are used on the area, i.e., with the agricultural machinery moving. "Offline" means that the machinery is not moving and/or the method is performed outside of the machinery and not yet interacting with the machinery.

A field or area contour is denoted by F1 in Figure 42. A geometrically translated lane in the same shape of the field contour is referred to as *perimetric*, outmost or headland lane, all expressions referring to the same, see F3 for visualization. Tree-island area contours are indicated by F2 in Figure 42. The geometrically translated lanes circling a tree-island are denoted as *island lanes* or circling island lanes; see F4 for illustration. Straight lanes F5 are designated *interior lanes* or straights. These lanes connect two opposite nodes of the perimetric lane F3 or a node along the perimetric lane F3 and along the island lane or circling island lane F4. Interior lanes do not necessarily have to be straight for the functionality of the concepts underlying the presented algorithms. Instead, interior lanes may in principle be of arbitrary shape, for example, curvedly aligned to a particular part of the field contour. However, straight lanes are preferred.

General Procedure P

Figure 45 explains the general framework from preparatory work via offline optimization to online tracking of the computed complete field coverage path. The first step, i.e. the preparatory work, is marked in Figure 45 with reference number S1, wherein reference number 1 refers to a step being part of this preparatory work. Step 1 involves the acquisition

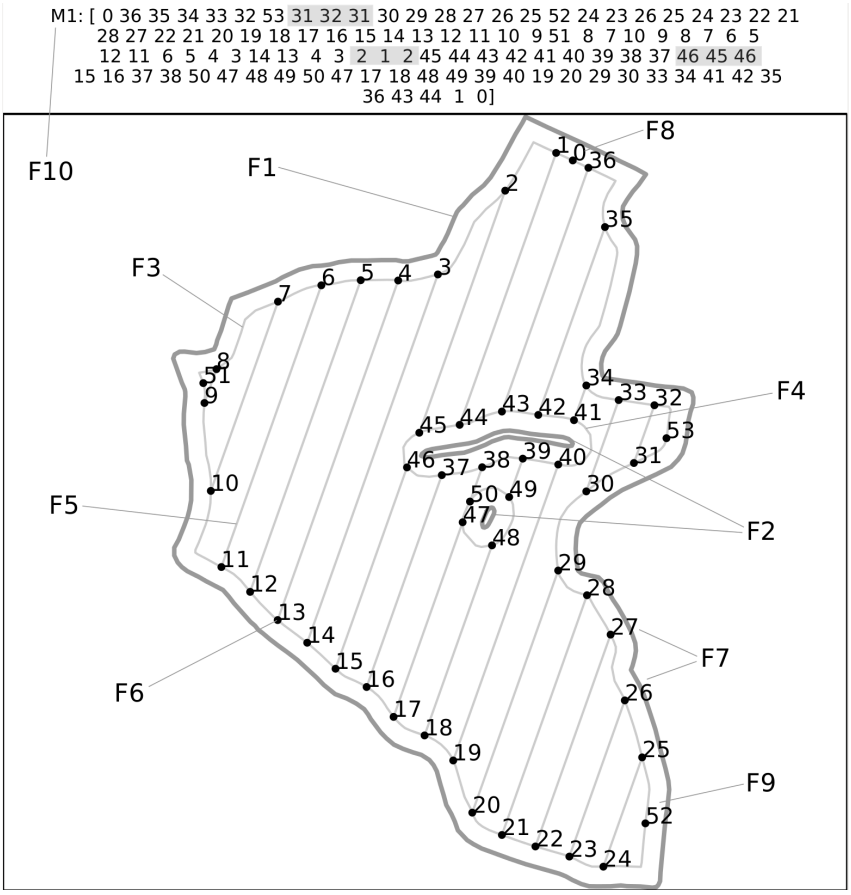


Figure 42: An offline-computed path as result of an optimization according to method M1 for an exemplary agricultural field including two interior tree-islands prohibited from trespassing.

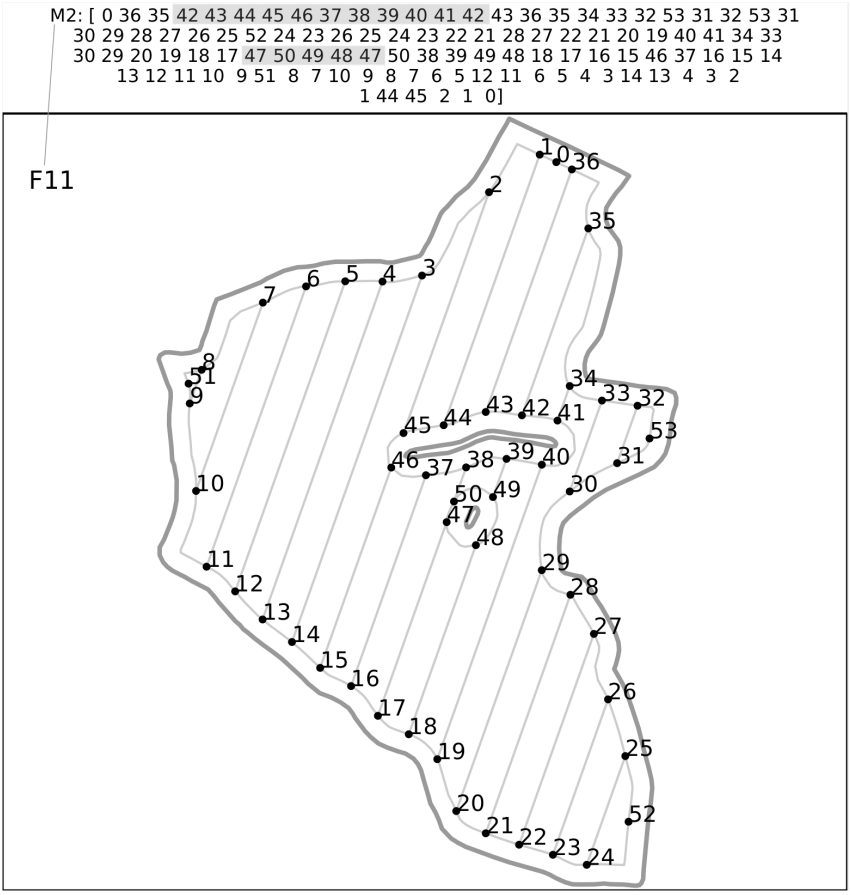


Figure 43: An offline-computed path as result of an optimization according to method M2 for an exemplary agricultural field including two interior tree-islands prohibited from trespassing.

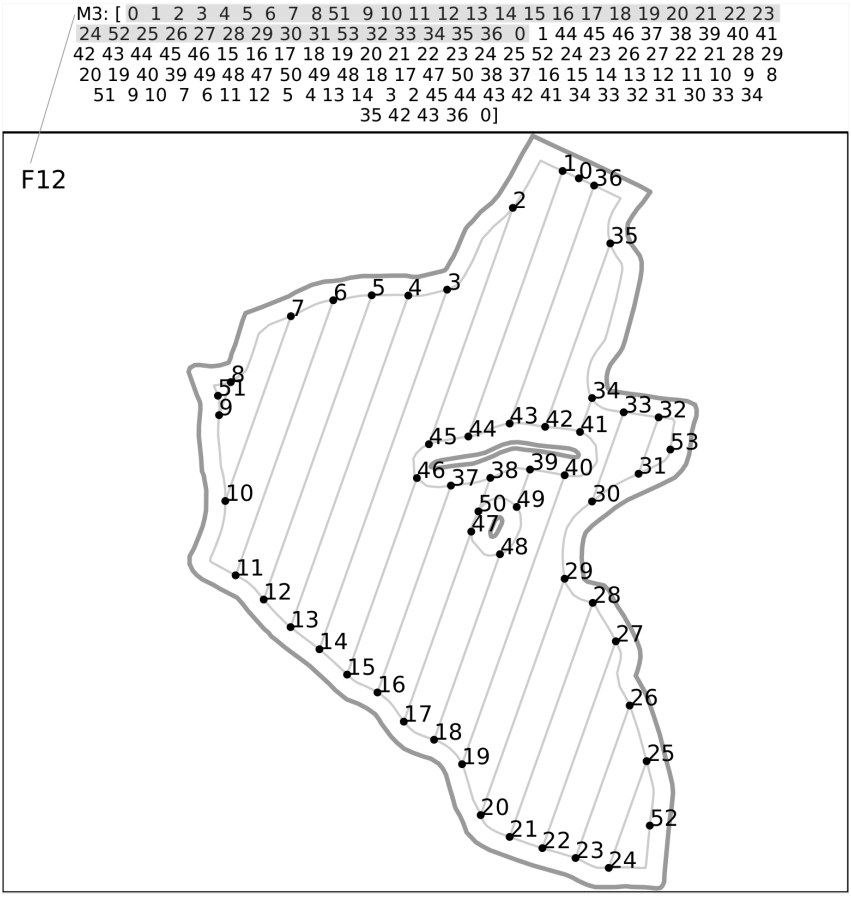


Figure 44: An offline-computed path as result of an optimization according to method M3 for an exemplary agricultural field including two interior tree-islands prohibited from trespassing.

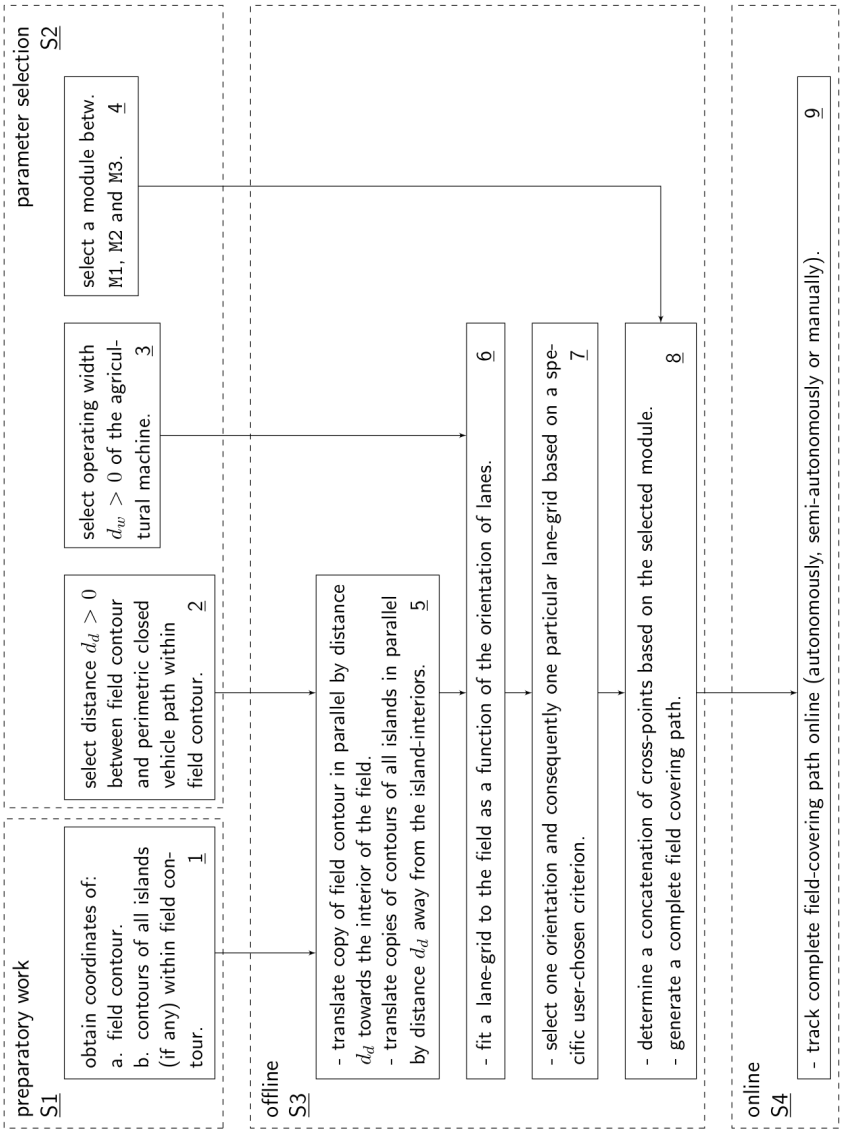


Figure 45: Flow of actions from field-specific data acquisition and user-parameter choices to online tracking of a computed field coverage path including optimization methods M1, M2 and M3.

of the position coordinates of the field contours of interest and, in case of existence, of the contours of all tree-islands.

The mode of this acquisition can be manifold. In descending order of accuracy, one method is to move along the field and tree-island contours while measuring and recording at appropriate distance intervals the positions by means of a location sensor. Alternatively, digital maps incorporating aerial and satellite photographs (for better orientation) may be used that allow the determination of, for example, latitude, longitude and altitude coordinates. Another option is the application of image analysis tools to aerial photographs, field records cadastral registers and the likes in combination with at least two reference points whose pixel- as well as GPS-coordinates are known, that allow an (approximate) conversion between image pixel-positions and the corresponding GPS-coordinates, e.g., expressed in the Universal Transversal Mercator (UTM) coordinate system. The disadvantage of the last two methods is that, for high-accuracy results, they require a free view on field and tree-island contours. Due to natural conditions such as shielding trees and the like this view may often be prohibited. Remedy can be provided by, instead of identifying the contour lines, identifying the (on images) already existing perimetric lane locally parallel to the field contour and likewise the tractor lanes locally parallel to the corresponding tree-islands, whose view is usually available. Otherwise, estimation, approximation and interpolation techniques can be employed. The disadvantage of the identification of the aforementioned already existing tractor lanes is that locally they may not be parallel to the field contours and thus incurring inefficiencies with respect to the usage of the available field area. Furthermore, photograph or image data may be outdated, blurred and noisy. For accuracy, the method described first, i.e., of recording positions while moving along the contours, is to be preferred if available.

As shown in Figure 45, a second Step S2, which is a parameter selection step, is performed which includes a selection of the operating width of the agricultural machine, denoted by $d_w > 0$, which also determines the distance between two parallel interior lanes, or the distance between locally parallel lanes in case of curvedly shaped lanes. This selection is

marked in Figure 45 with reference number 3. Reference number 2 refers to the step of selecting a distance between the field-contour and the locally parallel perimetric lane (and correspondingly for the tree-islands). This distance is denoted by $d_d > 0$ and is usually selected, for example, as $d_d = d_w/2$. Reference number 4 refers to the step of selecting one of three mathematical algorithms, hereafter also called methods: M1, M2 or M3. These algorithms will be explained in more details later in this text.

The third Step S3 is conducted offline based on the data acquired from the first Step S1 and the second Step S2, i.e., the parameter selection step. First, the perimetric lane and circling islands lanes (if islands are existent) are computed by the application of a geometrical translation technique, so-called *erosion* (mathematical morphological operation). This is shown in Figure 45 with reference number 5. A simple technique for its implementation is treating the contour lines as piecewise-affine line-fragments that can be geometrically translated by distance d_d in parallel. In a post-processing step, the resulting line is pruned to ensure keeping of a distance at least d_d to the field or island contours. Refinements of this method include preliminary interpolations and/or averaging. Consult also Figure 42 for visualization of the results, in particular, the field contour F1, the corresponding locally parallel translated closed perimetric lane F3, the tree- island contours F2 and their related locally parallel translated closed lanes F4.

Taking the operating width selection performed in Step 3 into account, a lane grid is fitted as a function of the orientation of interior lanes by intersecting the interior lanes with the eroded area contour. This step is marked in Figure 45 with reference number 6. This lane-grid in combination with the designated operating width ensures the coverage of the complete work area, i.e., the agriculturally used field. In the end, vehicles will move along these lanes.

In a further Step 7, an orientation and its corresponding lane-grid is selected. This step is a fundamentally important user-choice that usually already involves an optimization step preceding the subsequent minimization of the total coverage path length, in general, subject to constraints on the moving machinery. This is shown in Figure 45 with refer-

ence number 8. An orientation in Step 7 which may be chosen according to, for example, the following list of criteria:

- Minimization of the sum of the path lengths of all interior lane (usually straights) in between the perimetric lane and the closed island lanes (if existent); this criterion is of particular interest since it implies the minimization of repressed area which as a consequence is then available for additional planting.

- Minimization of the total field coverage path length. In contrast to minimizing the sum of the path lengths of only the interior lanes, perimetric and tree-island lanes that (potentially) are traversed multiple times for complete field coverage according to paths which have been optimized according to methods M1, M2 or M3, are now considered, too. In general, the corresponding orientation may be different for each of the three methods M1, M2, M3 and it may differ from the orientation associated with the minimization of only the sum of the path lengths of all interior lanes.

- Minimization of the total number of turns at headland.

- The avoidance of voltes and sharp turns, which may exclude particular orientations of the lane-grid.

- The avoidance of roll angles of the vehicle above a certain threshold resulting from field slopes in hilly terrain.

- The selection of a curved lane pattern aligned to a particular part of the field contour.

- Considerations about connections to roads, harvesting organization and logistics in general.

- Combinations and/or trade-offs of aforementioned criteria. For example, the combination of minimizing the sum of the path lengths of all interior lanes while simultaneously avoiding roll angles of the vehicle above a certain threshold resulting from field slopes.

The selection of the lane-grid implies the locations of the cross-points, i.e., positions where interior lanes and perimetric lane or circling tree-island lanes intersect. See F6 in Figure 42 for the visualization of such a cross-point. As illustrated by F7 in Figure 42, the cross-points are numbered. The starting and end position is assumed to be identical and is denoted by 0, see F8.

In Step 8, based on the previously determined lane-grid, all cross-points are concatenated efficiently minimizing the total field coverage path length taking one of the three methods M1, M2 or M3 into account. For an exemplary field with two tree-islands, F10, F11 and F12 illustrate the concatenations of cross-points resulting from one of the optimization methods M1, M2 and M3. Ultimately, by knowledge of the coordinate locations of all cross-points and linking lanes, the complete field covering path can be reconstructed, which completes the offline part visualized in Figure 45. The machinery has just to reach each of these cross-points in the order shown in F10, F11 and F12 in order to cover the area in the cost-optimized and most efficient optimized way.

Step S4 comprises the online tracking of the offline-generated field covering path computed in Step S3. This can be carried out by an autonomous vehicle, semi-autonomously or manually.

A possible implementation of subtasks or Steps 5, 6, 7 and 8, as indicated in Figure 45, is described below in procedure P, whereby details about the three specific optimization methods M1, M2 and M3. All of the three methods M1, M2 and M3 optimize the total area coverage path length; however, the methods used by each method M1, M2, M3 are subject to different constraints or heuristics.

The objective of the first method M1 is to minimize the total path length needed for the complete coverage of all edges associated with a transition graph and starting and ending at the same point (cross-point number 0, see Figure 42). Note that the connected undirected transition graph associated with a particular field is, throughout this section, always denoted by T_{field}. Eventhough heuristics are applied to avoid 180°-turns as often as possible, it is explicitly emphasized that the minimal total coverage path length associated with method M1 is always main-

tained. Thus, the shortest overall path length is never meant to be compromised or prolonged. Thus, method M1 gives the shortest path, but no guarantee on the avoidance of undesirable maneuvers such as, most importantly, 180°-turns. The complete avoidance of undesirable maneuvers such as 180°-turns can be achieved by the introduction of additional path-altering heuristics as done for the second and third optimization method M2 and M3.

The motivation for the introduction of the second optimization method M2 is to obtain a solution yielding a field coverage path length close to the minimum achieved by the first method M1, while at the same time always avoiding 180°-turns. This task is achieved by the admittance of particular heuristics that usually are to some very limited extent path length compromising. Two heuristics are described later in this text. Another difference, occurring only in case of the existence of tree-islands, refers to the particular method of making T_{M2} Eulerian by conducting a graph-augmentation yielding T^*_{M2}, see Algorithm 10.

The procedure P for the implementation of Steps $\underline{5}$, $\underline{6}$, $\underline{7}$ and $\underline{8}$ according to Figure 45 is described in Algorithm 8.

Three Methods: M1, M2 and M3

A particularity of both the first and the second method M1 and M2 is the coverage of the perimetric lane-segments on-the-fly. This implies, that the perimetric lane-segments do not have to be passed consecutively, i.e., by just driving once along the perimetric lane, before then starting to cover the straight lanes (interior lanes). Instead, lanes may be concatenated arbitrarily, in a total path length minimizing manner. In case there exist tree-islands, the circling island lanes are covered on-the-fly only for M1, whereas for M2 a heuristic is applied, see Algorithm 10.

A common guidance pattern employed at present by many agricultural vehicle drivers is initially to once drive along the perimetric lane before then working sequentially lane-by-lane in a typical *A-B pattern* (or similarly, first working the interior lanes before then ultimately circling the complete perimetric lane). The method M3 is motivated by this behavior. Thus, the heuristic of first always driving around the peri-

Algorithm 8: General Procedure P

1 Import location data of field and tree-islands contours.

2 Conversion to UTM-coordinates (to calculate in unit meters).

3 Parameter selection of d_d and d_w.

4 Determine perimetric and circling island lanes, see F3 and F4 in Figure 42.

5 **for** *a set of different lane-orientation angles* **do**

6 Coordinate system transformation (rotation of all available coordinates) to facilitate subsequent computations.

7 Computation of cross-points considering the d_w-parameter.

8 Labeling of cross-points (the start and end point for the field coverage path, i.e., the field entrance, is denoted by 0).

9 Establishing of lanes (edges) by connecting cross-point pairs.

10 Creation of a connected undirected transition graph T_{field}.

11 Evaluation of an optimization criterion (e.g., minimization of the sum of path lengths of all interior lanes between perimetric lane and the closed island lanes subject to the avoidance of roll angles of the vehicle above a certain user-defined threshold and the avoidance of voltes and sharp turns).

12 Selection of the optimal lane-orientation with respect to the chosen optimization criterion and constraints.

13 Storage of relevant information: lane-orientation, transition graph, as well as position coordinates of all cross-points and edges.

14 Computation of a field coverage path plan according to one or several of the three methods: M1, M2 and M3.

metric lane and thereby covering all of its associated lanes is incorporated in M3. However, in contrast to then following an A-B pattern, the third Algorithm M3 is used instead to plan the path minimizing the path length associated with the coverage of the remaining not-yet serviced interior lanes. See the highlighted cross-point sequence in F12 of Figure 44 for illustration of the initial circling of the perimetric lane. The third Algorithm M3 is intentionally designed such that any 180°-turns are always avoided. The heuristics described below are both used for the third method M3 when altering the Eulerian cycle associated with transition graph $\underline{T}_{\mathrm{M3}^*}$, see Algorithm M3. The two heuristics, i.e., the initial circling of the perimetric lane and the circling of tree-island lanes as soon as any cross-point associated with the corresponding tree-island is reached for the first time by the vehicle, naturally imply that for the coverage of the remainder of the field, the not-yet serviced interior lanes (usually straights) have to be concatenated efficiently in a least-cost manner by connecting them via some of the perimetric or tree-island lanes.

Figures 42, 43 and 44 illustrate results when performing the three different optimization methods M1, M2 and M3 for an exemplary field with two interior tree-islands. All of Steps S1, S2 and S3 mentioned above and shown in Figure 42 were conducted. For this particular case, the orientation and the consequent lane grid in block of Step 7 were chosen according to the criterion of minimizing the sum of the total path length of all interior lane-segments. The resulting cross-point sequences for the three methods of the three methods M1, M2 and M3 which define the pattern or path plan to be covered by the machinery are shown in lists F10, F11 and F12 being part of the corresponding Figures 42, 43 and 44.

Methods M1, M2, M3 are summarized in Algorithms (9), (10) and (12), respectively.

Algorithm 9: Method M1

1 Inputs: connected undirected transition graph T_{field}, position coordinates (spacing such that shape defining) of all edges of T_{field}, tilt angle of straight lanes, position coordinates of all cross-points, cross-point numbering, differentiation between perimetric and island-circling lanes.

2 Make T_{field} Eulerian to obtain T^*_{field}.

3 Determine an Eulerian cycle on Eulerian graph T^*_{field}, e.g., via the End-Pairing algorithm.

4 Apply heuristics to customize the Eulerian cycle:

- In general, for an Eulerian graph such as T^*_{field}, there does not exist only one unique corresponding Eulerian cycle.
- The first heuristic applied is the replacement of cross-point combinations such as a-b-a-m-n-b-c with a-b-n-m-a-b-c, which avoids the 180°-turn implied by the sequence a-b-a.
- The second heuristic applied as discussed below involving starting node 0.

5 Computation of the complete field covering path (from cross-points sequence to position coordinates associated with the Eulerian cycle).

6 Retransformation from rotated to original coordinate system.

7 Return values: Complete field covering path (e.g., UTM-coordinates), position coordinates of all cross-points, Complete field covering path expressed as a list of numbered cross-points (see F10 in Figure 42 for an example), and the field coverage path-length.

Algorithm 10: Method M2

1 Inputs: connected undirected transition graph T_{field}, position coordinates (spacing such that shape defining) of all edges of T_{field}, tilt angle of straight lanes, position coordinates of all cross-points, cross-point numbering, differentiation between perimetric and island-circling lanes.

2 Create undirected transition graph T_{M2}:

- T_{M2} is created as a copy of T_{field} after the dismissal of all circling island lane-segments (in case there are tree-islands present).
- For illustration, with respect to Figure 43, the dismissed circling bidirectional lane-segments comprise 37-38, 38-39, 39-40, 40-41, 41-42, 42-43, 43-44, 44-45, 45-46, 46-37, 47-48, 48-49, 49-50 and 50-47.
- Note that the resulting T_{M2} may not be connected: with respect to Figure 43, straight-segments 50-38 and 49-39 are not connected to the rest of T_{M2}.
- If there are no tree-islands, T_{M2} is just a copy of T_{field}.

3 Make T_{M2} Eulerian via a graph-augmentation yielding T^{*}_{M2}:

- Typically in the least-cost manner, however, under the explicit constraint of always ensuring connectivity of T^{*}_{M2}.
- With respect to Figure 43, a graph-augmentation of T_{M2} may include the pair-wise linking of odd-degree nodes 47-48, 50-49, 38-39, 40-41, 42-43, 44-45 and 46-37; however, this augmentation would result in the closed chain 50-49-39-38-50 being disconnected from the rest of the augmented graph, and is therefore prohibited.
- If there are no tree-islands, T^{*}_{M2} is always equal to T^{*}_{field}.

4 [Continued...]

4 Determine an Eulerian cycle on Eulerian graph T^*_{M2}, e.g., via the End-Pairing algorithm.

5 Apply heuristics to customize the Eulerian cycle:

- In general, for an Eulerian graph such as T^*_{M2}, there does not exist only one unique corresponding Eulerian cycle.
- A first heuristic applied is the replacement of cross-point combinations such as a-b-a-m-n-b-c with a-b-n-m-a-b-c, which avoids the 180°-turn implied by the sequence a-b-a.
- A second heuristic applied as discussed below involving starting node 0.
- A third heuristic applied is motivated by avoiding 180°-turning, involving the utilization of auxiliary cross-points; this heuristic is path length-altering.

6 Alteration of the Eulerian cycle associated with T^*_{M2} by the inclusion of circling island lanes:

- The result of this step is referred to as the *altered Eulerian cycle* associated with T^*_{M2}; the original Eulerian cycle does not cover all lanes associated with T_{field}, however, the altered Eulerian cycle now does; in particular, in a way that is meant to ensure avoidance of any 180°-turns.

7 Computation of the complete field covering path (from cross-points sequence to position coordinates associated with the altered Eulerian cycle).

8 Retransformation from rotated to original coordinate system.

9 Return values: Complete field covering path (e.g., UTM-coordinates), position coordinates of all cross-points, Complete field covering path expressed as a list of numbered cross-points (see F11 in Figure 43 for an example), and the field coverage path-length.

Algorithm 11: Method M3

1 Inputs: connected undirected transition graph T_{field}, position coordinates (spacing such that shape defining) of all edges of T_{field}, tilt angle of straight lanes, position coordinates of all cross-points, cross-point numbering, differentiation between perimetric and island-circling lanes.

2 Create undirected transition graph T_{M3}:

- T_{M3} is created as a copy of T_{field} after the removal of all perimetric lane-segments; likewise, in case there exist tree-islands, all circling island lane-segments are dismissed from T_{field}.
- Then, T_{M3} consists of only unconnected interior lanes (usually straights).

3 Extend T_{M3} to a connected undirected graph $\underline{T}_{\text{M3}}$:

- To make T_{M3} connected, lane-segments are added to it.
- Added lane-segments are elements of only the perimetric lane or the circling tree-island lanes.
- Lane-segments are added in a least-cost manner while simultaneously avoiding the formation of any loops with the lane-segments already chosen until the resulting graph $\underline{T}_{\text{M3}}$ is connected (see [233] for mathematical background).

4 Make $\underline{T}_{\text{M3}}$ Eulerian via a graph-augmentation yielding $\underline{T}^{*}_{\text{M3}}$:

- Typically conduct the graph-augmentation of $\underline{T}_{\text{M3}}$ in the least-cost manner under explicit constraint of always ensuring connectivity of the resulting $\underline{T}^{*}_{\text{M3}}$.

5 [Continued...]

5 Determine an Eulerian cycle on Eulerian graph $\underline{T}^*_{\text{M3}}$, e.g., via the End-Pairing algorithm.

6 Apply heuristics to customize the Eulerian cycle (for Eulerian graph $\underline{T}^*_{\text{M3}}$ there does not exist a unique corresponding Eulerian cycle).

7 Alteration of the Eulerian cycle associated with $\underline{T}^*_{\text{M3}}$ by the inclusion of circling island lane-segments and the initial perimetric lane circling:

- The direction of the initial perimetric lane circling is selected such that a smooth transition to the next cross-point on the path is ensured avoiding any 180°-turn; for illustration, see the highlighted part of F12 in Figure 44 and its fluid progression (avoiding any 180°-turning) to the remainder of the field coverage path.
- See the description for details about the inclusion of circling island lane-segments.
- The result of this step is referred to as the altered Eulerian cycle associated with $\underline{T}^*_{\text{M3}}$; the original Eulerian cycle does not cover all lanes associated with T_{field}, however, the altered Eulerian cycle now does; in particular, in a way that is meant to ensure avoidance of any 180°-turns.

8 Computation of the complete field covering path (from cross-points sequence to position coordinates associated with the altered Eulerian cycle).

9 Retransformation from rotated to original coordinate system.

10 Return values: Complete field covering path (e.g., UTM-coordinates), position coordinates of all cross-points, Complete field covering path expressed as a list of numbered cross-points (see F12 in Figure 44 for an example), and the field coverage path-length.

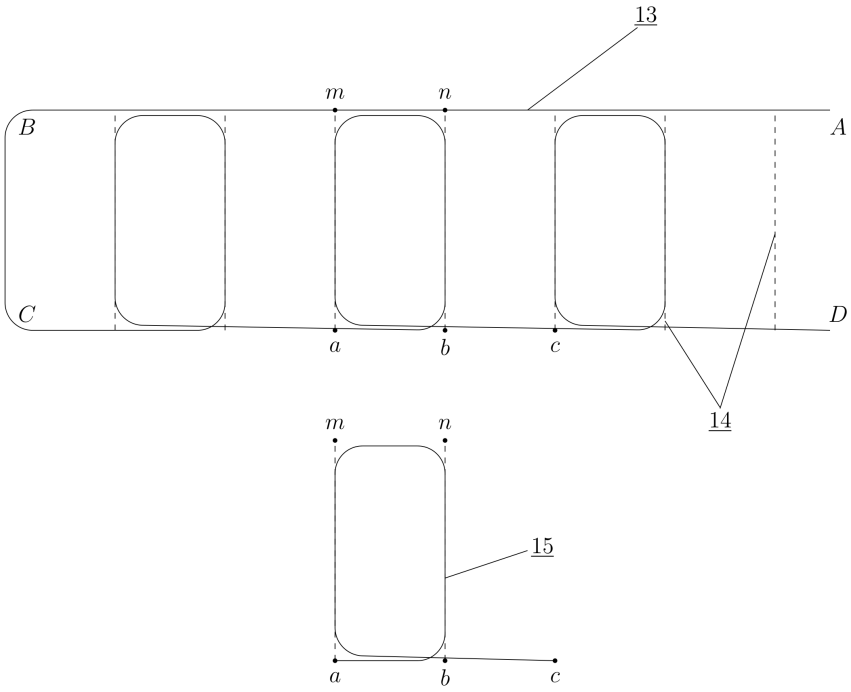


Figure 46: A guidance pattern naturally occurring as a result of an optimization according to either one of methods M1 and M2, wherein the field coverage path is intentionally not aligned with straight lanes for better visualization of the underlying cross-points sequence logic and path plan.

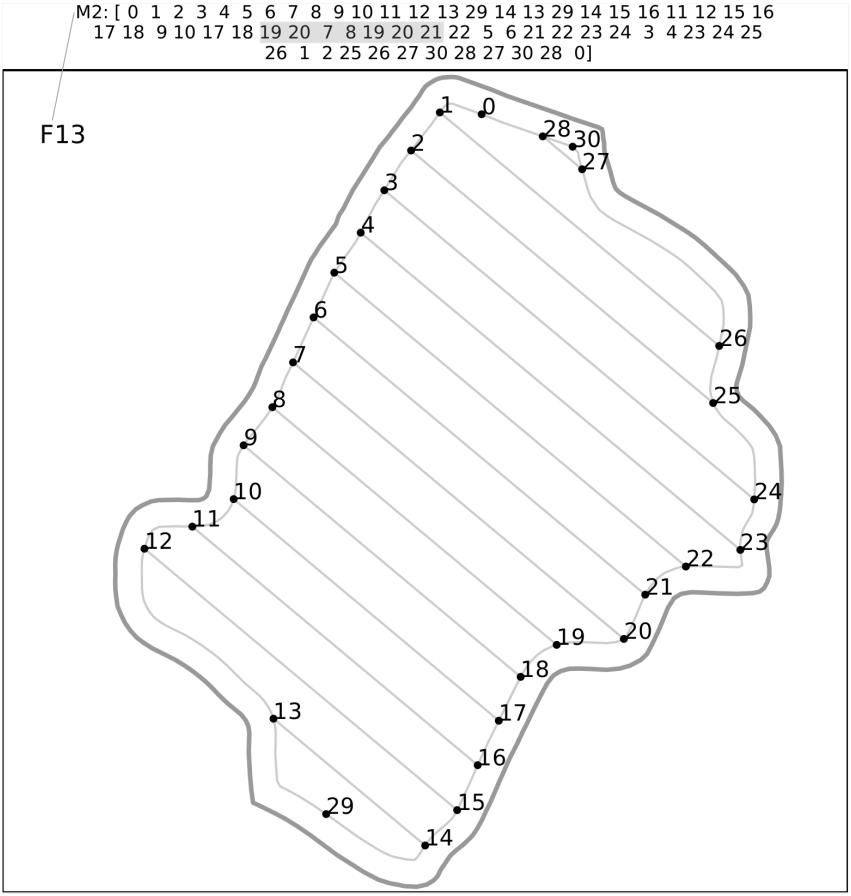


Figure 47: An offline-computed path as result of an optimization according to method M2 for an exemplary agricultural field without interior tree-islands prohibited from trespassing.

A guidance pattern naturally occurring as a consequence of the first and second optimization method M1 and M2 is sketched in Figure 46. This circular driving framework is of particular interest since it not only avoids 180°-turns and is thus simple to drive, but more importantly prescribes how to, in a minimum total path length-sense, optimally cover also rectangular or, more generally, regularly shaped fields. Figures 47 and 48 emphasize the natural occurrence of this particular guidance pattern as a result of method M2- in two real-world settings with naturally shaped field contours. F13 and F14 illustrate the optimal concatenations of cross-points according to method M2. The minimum total length field coverage path 13 in Figure 46 is composed of passing through the headland indicated by reference points A, B and C before then working all straight lanes, denoted by 14, in a specific circular manner (marked with reference number 15 in Figure 46), always comprising two straight lanes, until point D is reached. The specific sequence of cross-points for the circular pattern reads a-b-n-m-a-b-c. The reason for optimality of the conjunction of headland path A-B-C and subsequent multiple circular guidance patterns is the coverage of the bottom headland lane (or perimetric lane) on-the-fly. For total path length-optimality, a key is the appropriate pairing of cross-points such that the sum of all lane-parts covered twice, such as a-b and m-n etc., is minimal. Ultimately, as a detail, due to the particular artificial and consecutive numbering of cross-points illustrated, cross-point combinations a-b-c are usually (but not necessarily) increasing, e.g., 15-16-17, 18-19-20 etc. as shown in F13 of Figure 47, or decreasing, e.g., 18-17-16 or 16-15-14 etc. as displayed in F14 of Figure 48.

Note that the circular pattern sketched in Figure 46 may potentially include circling of islands as well. Consider F14 in Figure 48 and its highlighted cross-point sequence ...-8-7-41-44-43-42-41-44-34-33-43-42-8-7-6-... which includes the particular circular guidance pattern 8-7-34-33-8-7-6 interrupted by the circling of the tree-island 41-44-43-42-41. It is emphasized that the given concatenation of cross-points is enforced by the design of the algorithm, i.e., the tree-island is made Eulerian along the straight lane-segments. This means that cross-points 41 with 44 and

42 with 43 are connected during the Eulerian-step of method M2. The Eulerian-step of method M2 is always conducted such that it ensures reachability of all cross-points and thus coverage of the complete field. Suppose the two tree-islands in Figure 43 are made Eulerian by duplicating cross-point pairings 47-48, 49-50, 38-39, 40-41, 42-43, 44-45, 46-37 which for the given field under consideration would be path length-optimal, the quadrilateral cycle 50-49-39-38-50 would then be created disconnected from all other cross-points after the Eulerian-step.

As mentioned in the Procedure P of the framework, auxiliary cross-points (nodes) were introduced to ensure unique connections between any two cross-points. With respect to Figure 49, such auxiliary inserted cross-points are numbered 21, 22 and 23. Consider the case of cross-point 23. Without its existence, the connection between cross-points 18 and 19 would not be uniquely defined. These auxiliary cross-points are also of particular interest for method M2. Suppose the edge 18-19 is element of the list of pairings that make transition graph T_{M2} Eulerian. Then, the edge 18-19 will be passed through twice. This fact, in combination with cross-points 18 and 19 each having only one non-auxiliary cross-point neighbor, implies the shortest path for coverage of the 18-19 edge to be the cross-point sequence 18-19-18. As shown in Figure 49, see F15, this sequence occurs naturally in the M1-solution. The disadvantage of this sequence is, however, the 180°-turn. For this practical reason, cross-point sequences such as 18-19-18 are therefore in method M2 always replaced by utilizing the neighboring cross-point. Thus, a sequence of 18-19-18 is in the second method M2 replaced (in the heuristics step), see Algorithm 10, by the sequence 18-19-23-18, for illustration highlighted in F16 of Figure 49. This heuristic is also used for the method M3, for example, see F12 in Figure 44, where the 180°-turn implied by 24-25-24 is altered to 24-25-52-24.

Suppose the starting and end node (entrance and exit to the field), always denoted as cross-point 0, lies on a lane-segment belonging to the list of pairings that make a connected undirected transition graph T Eulerian. For illustration, consider Figure 49, where the odd vertices 20 and 1 are paired via path 20-0-1 and belong to the list of pairings that make T

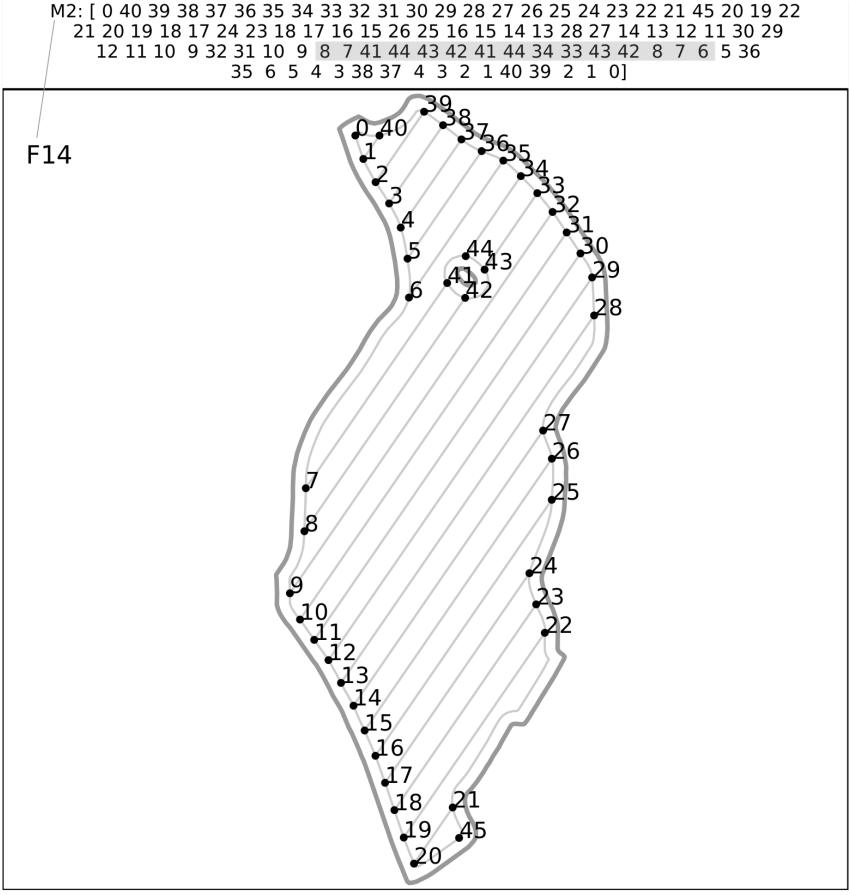


Figure 48: An offline-computed path as result of an optimization according to method M2 for an exemplary agricultural field including one interior tree-island prohibited from trespassing.

Eulerian. Then, for the Eulerian tour, the undirected edges 20-0 and 0-1 must each be passed twice. In such a case, by, for example, the application of the End-Pairing algorithm described in [101], the last seven cross-points of a valid Eulerian tour may be 1-16-17-20-0-1-0, which imply the 180°-turn 0-1-0. To avoid this 180°-turn, a heuristic is motivated which always replaces sequences for the last cross-points of an Eulerian tour in the general form of x-k-l-m-0-x-0 by x-0-m-l-k-x-0. This does not alter the path length, but just the sequence of covered cross-points. This heuristic is therefore not only applicable to method M2 but also to method M1. With respect to Figure 49, by the application of the discussed heuristic, 1-16-17-20-0-1-0 is transformed to 1-0-20-17-16-1-0 as indicated in F15 and F16 for the corresponding first and second methods M1 and M2.

Another heuristic follows the concept of circling a tree-island as soon as any cross-point associated with that tree-island is reached for the first time. This heuristic is employed by both methods M2 and M3. For illustration, consider Figure 43 and the cross-point sequence 0-36-35-42-43 which is part of an Eulerian cycle associated with T^*_{M2}. Cross-point 42 belongs to the circling tree-island lane described by cross-points {37, 38, 39, 40, 41, 42, 43, 44, 45, 46}. The strategy is then to incorporate these cross-points into the Eulerian cycle associated with T^*_{M2} and thus obtain 0-36-35-42-43-44-45-46-37-38-39-40-41-42-43-... as indicated in F11 of Figure 43. Note that the order of the added circling tree-island cross-point sequence is always chosen such that a smooth transition to the next cross-point on the path is ensured avoiding any 180°-turn. Consider for visualization the second tree-island in Figure 43 given by the cross-point set {47, 48, 49, 50}. The cross-point sequence 19-18-17-47-50 is part of the Eulerian cycle associated with T^*_{M2} and the second tree-island is reached for the first time at cross-point 47. Thus, according to the strategy and since, according to the Eulerian cycle, the cross-point following 47 is 50, we round the tree-island clock-wise and obtain the updated path 19-18-17-47-50-49-48-47-50-... as indicated in F11 of Figure 43. Ultimately, it is pointed out that the strategy described in this paragraph is explicitly coordinated with the way how the undirected graph T_{M2} is created, see Algorithm 10. The fundamental motivation is the purpose, of ensuring

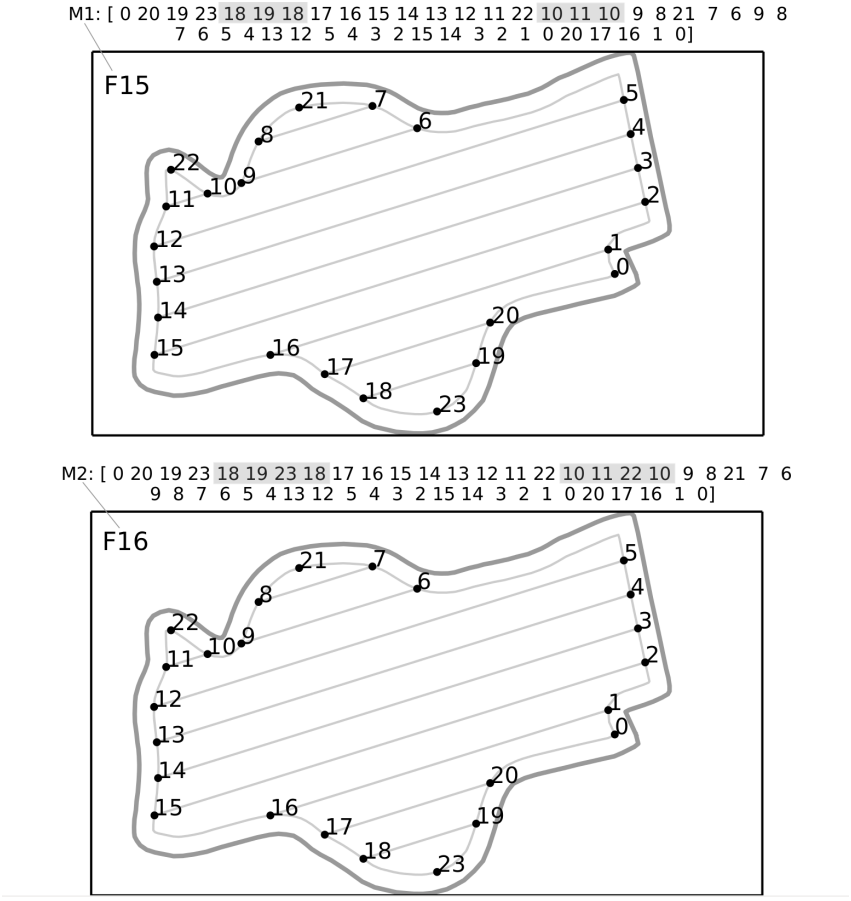


Figure 49: The difference between the results obtained when applying optimization according to methods M1 and M2 to an exemplary agricultural field not including any interior tree-islands prohibited from trespassing.

180°-turns are avoided even if tree-islands are existent in the interior of the work area.

Figure 49 compares the typical results of methods M1 and M2 for an exemplary field not including any interior tree-islands prohibited from trespassing. The cross-point sequence corresponding to method M1 is shown in F15, and the result associated with method M2 in F16. The differences between the two solutions are highlighted. In case of method M2, the path is altered according to the logic described herein in order to avoid the two 180°-turns characteristic for the first method M1. Besides this, there is no difference in path, which is conceptually always the case if no tree-islands are existent in the interior of the work area.

Figure 50 is meant to further illustrate the suitability and characteristics of a typical field coverage path, see F17, returned by method M2 for an exemplary agricultural field including multiple interior tree-islands prohibited from trespassing. See also Figure 51.

The turning trajectories and dynamics of the vehicle at headlands or in general are not taken into account explicitly in the drawings, see, e.g., Figure 47. This is also due to the fact of the applicability of the present method to a great variety of vehicles, scales and shapes of closed fields. The omission of precise turning trajectories does not violate functionality of the devised algorithms in any sense. An inclusion and description of precise turning trajectories is to be considered as a more detailed modeling of the lane-grid fitted to the particular field-shape and may be of specific importance in Step 7 of Figure 45, when selecting the orientation of the lane-grid, for example, according to the criterion of avoiding voltes and sharp turns.

The methods for the computation of vehicle paths for complete field coverage are also of interest in combination with variable rate applications for, e.g., spraying and fertilizing. The tracking of offline-computed and thus known paths can be used as feedforward information for the variable rate application controllers. Alternatively, this information can also be used for vehicles not equipped with expensive automated variable rate application apparati in the sense that in form of visual and audio support the human vehicle driver can be signaled where and when to

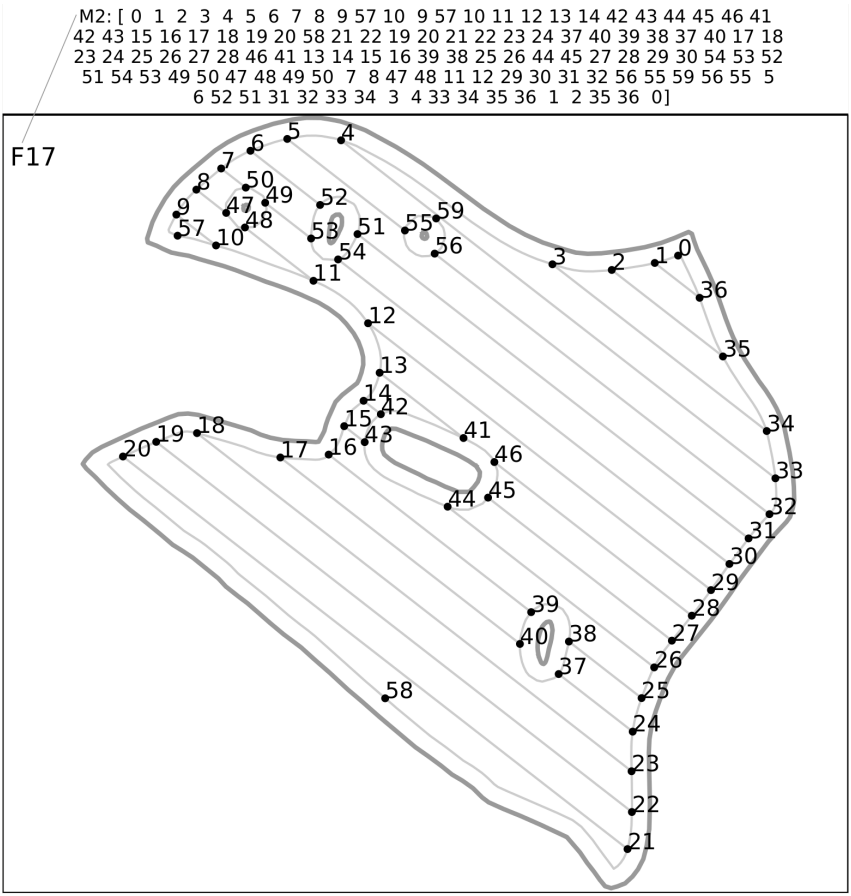


Figure 50: An offline-computed path as result of an optimization according to method M2 for an exemplary agricultural field including multiple interior tree-islands prohibited from trespassing.

Figure 51: Exemplary field. The blue line shows the optimized trajectory according to the second method from [146], see Figure 50. Optimization criteria are a) *repressed area minimization*, and b) consequent minimal path length edge cover. Repressed area here refers to field areas repressed by tractor tires. These areas are consequently not usable for plantation. The original path (used by farmers) is indicated by the white background.

switch off, e.g., spraying and fertilizing machinery when covering a particular lane-segment, for example, for the second time. The switching-information provided to the user can be further refined when taking dynamic mathematical models of vehicle, trailer and, e.g., spraying machinery into account, for example, for the modeling and prediction of dynamical response times.

Figure 52 illustrates an efficient overlap-avoiding on/off switching scheduling of fertilizer spreaders or spraying application machineries that are mounted on a vehicle system and operating perpendicularly to the vehicle direction of travel with a specific operating width. Specifically, the scheduling is tailored to the path planning methods for field coverage presented above. The fundamental objective is the avoidance of fertilizing or spraying overlap, i.e., the avoidance of additional application of fertilizers, sprays or similar substances on areas already served, when moving along a path plan for field coverage as, e.g., devised by method M2 and as further illustrated in Figure 46. The operating width of the agricultural machinery is marked with reference numbers 20 and 18, here referred to as field contour and *headland application bound*, respectively. On and off switching of fertilizer spreaders or spraying application machineries are indicated by circular and square markers, having reference numbers 25 for "on" and 24 for "off", respectively. It is distinguished between, first, path segments with switching on (application of fertilizers or sprays) as illustrated by thickened line 19 or 23, and, second, path segments with switching off (no application of fertilizers or sprays) illustrated by thinner lines as in 21. It is further distinguished between, first, switching decisions to be taken at intersections of interior lanes and application bounds such as indicated by 17, 22, and, second, switching decisions to be taken at positions 24 and 25, both located along headland lanes. This distinction is required as a result of the devised path planning methods for field coverage, in particular according to method M2. It is further different to existing sequential lane-by-lane operation where efficient overlap avoidance is already achieved by switching at only the intersections of interior lanes and application bounds (and not additionally along headland path segments), however, at the cost of an increased

path length stemming from a continuous headland path coverage, typically before the coverage of any interior lanes, or, alternatively, after the coverage of all interior lanes. In contrast, characteristic to method M2 is a shortened path length, however, at the cost of requiring two more switching decisions per every two interior lanes, as visually exemplified in Figure 52.

An efficient switching scheduling for overlap avoidance, as further visualized in Figure 52, has to be applied anywhere in between the end and the beginning of turning maneuvers of, first, traversing from an interior lane towards a headland lane as illustrated by reference number 26, and, second, traversing from a headland lane towards an interior lane as illustrated by 27, respectively. With respect to the traversal pattern from Figure 46, i.e., a-b-n-m-c, as further visualized in Figure 52, the efficient switching sequence for overlap avoidance is summarized as follows.

Algorithm 12: Switching Sequence for Overlap Avoidance

1 With regard of Figure 52, come from node a with switching on;
2 Switch off at node 28;
3 Continue driving (passing node b) until reaching the next intersection of application bound and interior lane where switch on;
4 Continue until the next intersection of application bound and interior lane where switch off;
5 Continue driving (passing node n and m) until the next intersection of application bound and interior lane where switch on;
6 Continue driving, and switch off at the next intersection of application bound and interior lane;
7 Continue driving (passing node a) until reaching of location 28 where switched on;
8 Continue to drive until node c while switched on.

Finally, the headland lane coverage in Figure 52 between node n and m is cluded when initially, and typically for method M2, covering more than half of the headland path with switching on. As illustrated in Figure 52, this occurs after the traversal from 16 until the first off-switching at position 24.

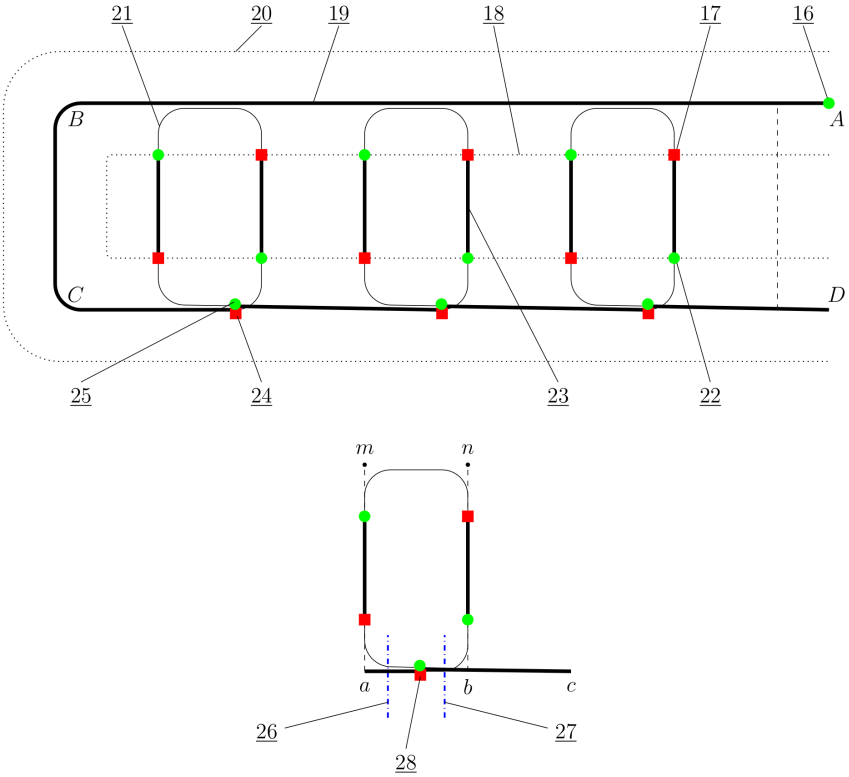


Figure 52: An efficient on/off switching scheduling of a fertilizing or spraying application machine mounted on a vehicle system moving along the path plan according to Figure 46.

3.1.3 Quantitative Example

For the exemplary field visualized in Figures 50 and 51, quantiative savings are summarized in Table 5. Two comments are made. First, note that results are reported for a *single* field coverage. For a complete crop-related work-cycle, e.g., a yearly work-cycle for rapeseed or wheat in Northern Germany, up to 10 field coverages may be required per work-cycle. Second, results are reported for a *single* field. A large farm or collaboration of multiple smaller farms may have to service up to hundreds of fields. The total saving potential has to be interpreted accordingly.

Table 5: Abbreviations ASPL and ATPL refer to *accumulated straights path length* and *accumulated total path length*, respectively. A reduction in ASPL implies that a more efficient field coverage path could increase plantable area by reduction of unnecessarily repressed area through tractor tires. Monetary loss due to unnecessarily repressed area by tractor traces can be computed as the product of the total length of repressing tractor traces, the tire width, and the normalized gain for the corresponding field crop. Results are reported for the field in Figures 50 and 51. The *current* and optimized field coverage path plans are compared, whereby method M2 is employed for optimization. Savings are reported in [m] and [%].

	Current	Optimized M2	Saving [m]	Saving [%]
ASPL	7950m	7672m	-278m	**-3.5**%
ATPL	14934m	13308m	-1626m	**-10.9**%

3.1.4 Hierarchical Controller Parametrization

Offline

For the *offline* planning of a field coverage path plan there are two main layers. These are:

1. General procedure P: The selection of an optimal lane-orientation and consequent fitting of a lane-grid to the field of interest.

2. One of methods M1, M2 or M3: The computation of a complete field covering path based on the transition graph associated with the fitted lane-grid from layer 1.

Online

For the *online* implementation of the field coverage path there is one main layer. This is represented by a vehicle system moving along the path plan (for example, automated according to the method from [152]), and simultaneously executing the switching method according to Algorithm 12 for the actuation of fertilizing or spraying application machines mounted on the vehicle system.

3.2 In-Field Navigation via an Android App

This section summarizes [147]:

- M. Graf Plessen, "System and method for navigation guidance of a vehicle in an agricultural field," *WO Patent App. PCT/EP2016/072968*, June 8 2017.

In order to realize a navigation device for the method from Section 3.1, a simple data based Android App was written accessing smarthpone GPS-sensors. Thereby, it was found that low-cost GPS-sensors in smartphones are too inaccurate and disctracting (due to "jumping" position indicators) for providing in-field navigation. Therefore, another Android app was designed based solely on human feedback for localization [147], see Figures 53 and 54.

3.2.1 Background

Currently, navigation guidance of agricultural machinery within agricultural fields is served in combination with auto-steering systems based on actuators, e.g., for controlling the steering angle and vehicle speed, the corresponding control algorithms, and, specifically important, accurate location sensors such as satellite-based positioning systems (e.g., GPS receivers with differential correction). In alternative to these fully autonomously operating systems, there exist semi-autonomously operating systems, where headland turns need to be conducted manually by a human driver, before lining up the vehicle with the next lane to be driven, and before engaging auto-drive for the traversal of the (typically straight) lane ahead until the next headland turn, which is then again to be conducted manually.

All semi-autonomous and fully autonomous steering systems require high-precision location sensors with positioning accuracies in the low centimeter range. Location sensors may return measurement failures. This may be due to shielding trees, low-hanging branches which disable satellite communication or similar environmental conditions, or other hazards that might be harmful to the desired operation of the location

sensors. For safety reasons and by law, existing industrial auto-steering systems as well as semi-autonomous steering systems require a human operator to constantly supervise machine operations.

3.2.2 Summary

In [147] a system and method are described for enabling navigation guidance of a manually driven vehicle in agricultural fields without the usage of high-precision location sensors such as GPS receivers with differential correction. A human operator of the vehicle interacts at preplanned waypoints recursively with a supporting mobile computing device (e.g., a handset), thereby obtaining navigation instructions for the following of a predetermined path plan for field coverage or a similar path plan for traversal within agricultural fields, see Figure 53. The predetermined path plan for field coverage is stored in a database in form of geographic location coordinates to which the mobile computing device has access. Location is thus implemented by the human operator orienting himself within the agricultural field by aid of a mobile computing device with audio and visual support. Navigation is thus implemented by the recursive interaction between the human operator and the supporting mobile computing device via a user interface that is available to the operator.

3.2.3 Web-service for Communication

For the system and method for navigation guidance of a manually driven vehicle in agricultural fields without the usage of artificial location sensors as described above, a network interface and a location sensor are not necessary. However, the device may be equipped with both a (limitedly accurate) location sensor and a network interface. In this case, multiple of mobile computing devices can communicate with each other via a service, typically a web-service.

Figure 54(a) illustrates an example of a user interface that is displayed to a human supervisor or coordinator of farm operations. A location sensor enables the display of the coordinators geographic location within a map, together with displays of, for example, agricultural field con-

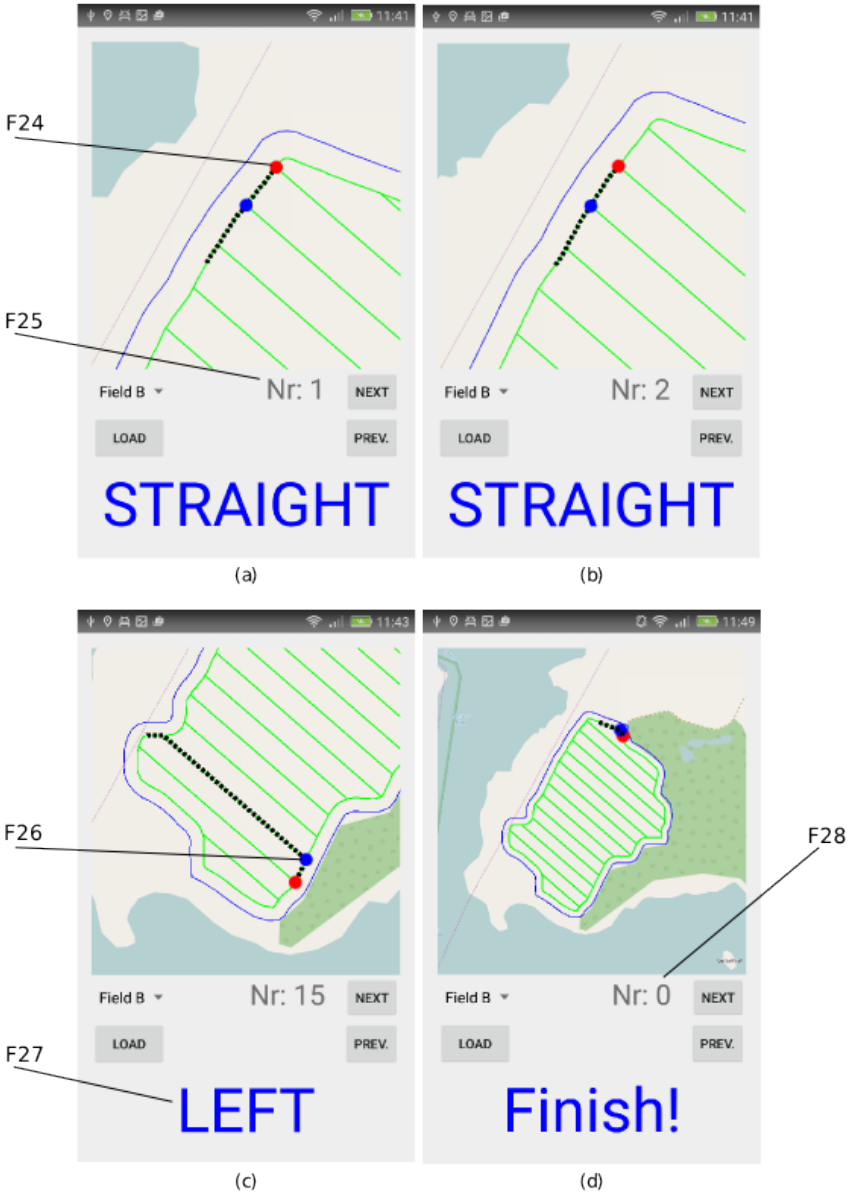


Figure 53: Snapshots of navigation instructions returned by a simple Android App according to [147].

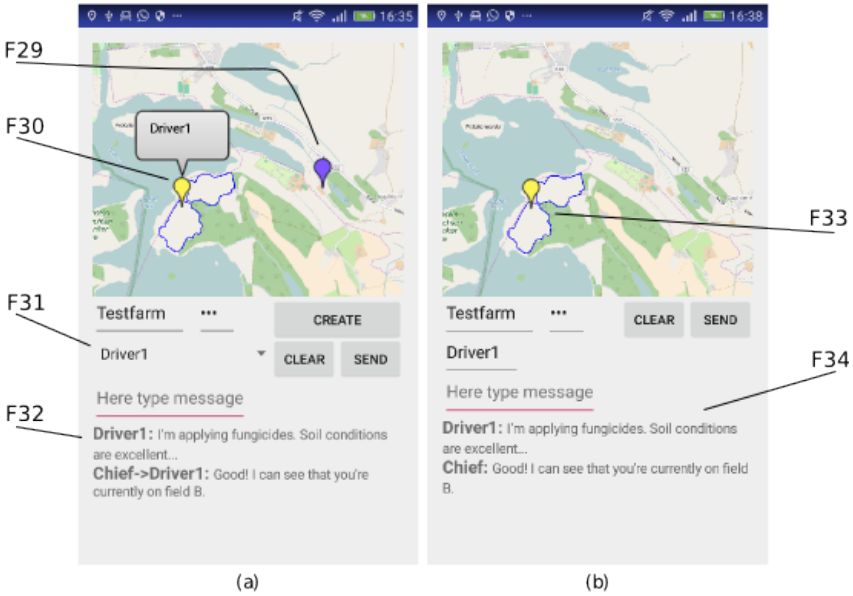


Figure 54: Snapshots of (a) a user interface that is displayed to a human supervisor or coordinator of farm operations, and (b) a user interface that is displayed to an operator of a mobile agricultural machinery. Communication is enabled by a web-service [147].

tours of interest. A network interface enables to also inform about geographic location information of multiple supervised persons, for example, multiple different operators of multiple different mobile agricultural machines. One supervised person in Figure 54(a), labelled as Driver 1, is therefore representative. A network interface allows for communication between supervisor and supervised operators in form of, for example, message exchanges as illustrated. This may be, for example, for the purpose of abortion, rescheduling or alternation of field coverage path plans due to unforeseen events or unpredicted and uncontrollable influences such as, for example, sudden changes in weather conditions. Figure 54(b) illustrates an equivalent example of a user interface that is displayed to an operator being supervised, typically, the operator of a mobile agricultural machinery.

The repeated and memory-aware tracking of received feedback from the human operator about the last covered intersection points enables the monitoring of progress in the realization of field coverage path plans or similar path plans for traversals within agricultural fields. This progress may be displayed in real-time to the operator of the vehicle system via a device and a user interface. In addition, network access and repeated and memory-aware tracking of received feedback from the human operator about the last covered intersection points, and its transmission via a service, typically a web-service, to a supervising party, enabled by the network interface, enables the supervisor or coordinator of field operations the monitoring of progress of mobile agricultural machines in the realization of field coverage path plans or similar path plans for traversals within agricultural fields.

3.2.4 Hierarchical Controller Parametrization

The control layer is represented by the Android App, which accesses a) a database, and b) possibly a web-service.

3.3 Partial Field Coverage Based on Two Path Planning Patterns

This section summarizes [145]:

- M. Graf Plessen, "Partial field coverage based on two path planning patterns," draft at *arXiv preprint arXiv: 1707.07649*, 2017.

A method for *partial* area coverage is presented. Lighter machinery with smaller storage tanks can alleviate soil compaction, but does not permit to cover a given field area in a single run, for example, during a spraying application. Instead, multiple returns to a mobile or stationary depot located outside of the field are required for storage tank refilling. Therefore, a suitable path planning method is suggested that accounts for the limited turning radii of agricultural vehicles, satisfies repressed (compacted) area minimization constraints, and aims at overall path length minimization. An interesting finding, and ultimately also the motivation for [145], was that the paths returned from the method according to [146] are a naturally, and often optimally, good fit for the partial planning setting, too. This holds specifically for convexly shaped fields.

3.3.1 Motivation

According to [2] and [53], the agri-food supply chain can be decomposed into four main functional areas: production, harvesting, storage and distribution. For improved supply chain efficiency, logistical optimisation and route planning play an important role in all of the four functional areas. Regarding production, for example, by means of minimization of the non-working distance travelled by machines operating in the headland field [62], optimal route planning based on B-patterns [61], or route planning for the coordination of fleets of autonomous vehicles [83], [335]. See also [90] for an overview of means for efficiency improvements, [54] for the importance of satellite-based navigation systems in modern agriculture, and [345] for a distinction between in-field, inter-field, inter-sector and inter-regional logistics. The presented path planning method for

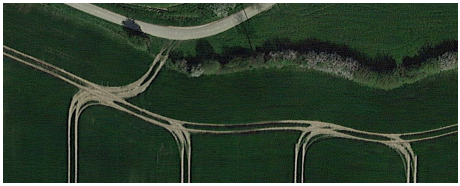


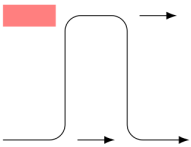
Figure 55: (Left plot) Visualization of real-world transitions between headland path and interior lanes. In the satellite image, the effects of a limited turning radius of the employed agricultural vehicle are visible. The repressed areas are indicated by the bright tire traces. Note also the repressed area due to the connection between field entrance and headland path. (Right plot) Visualization of the *repressed area minimization constraint*.

partial field coverage relates to the first functional area of the agri-food supply chain: production.

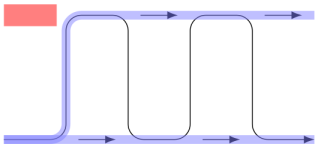
The last decades have witnessed a trend towards the employment of larger and more powerful machines in agriculture. This trend is expected to further continue in the near future, see [236] and [89]. Among the main benefits are higher work rates. The drawbacks include increased soil compaction due to machinery weights, see [319] and [171]. See also [12] for the influence of tire sizes on soil compaction. Concurrently to this ongoing trend, there are alternative considerations about the replacement of heavy machinery by teams of smaller and lighter autonomous robots to mitigate soil compaction, see [51], [57], [58], [54], [140] and [335]. See also [366] for a method for motion coordination of teams of autonomous agricultural vehicles.

Motivated by the concept of smaller collaborating machines, a path planning method for *partial* field coverage is presented, which is characterized by a) minimization of traveled non-working path length, and b) compliance with repressed area minimization constraints. The latter implies driving along unique and established transitions between headland path and interior lanes, thereby avoiding the creation of any additional tire traces that result from vehicle traffic passing over crops and compacting soil. Two different path planning patterns are discussed.

In contrast to route planning methods such as in [84] for the in-field



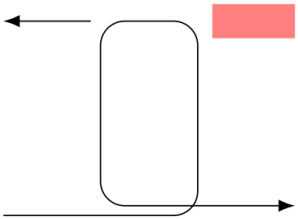
(a) The path planning pattern for ABp.



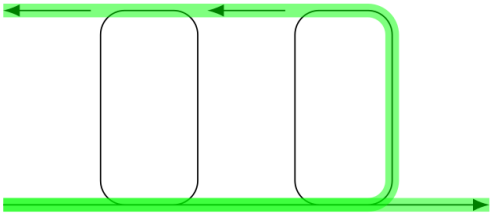
(b) Concatenation of two path planning patterns for ABp.

Figure 56: (Top plot) Illustration of the first path planning pattern. The red bar indicates the area that cannot be reached by neither traversal of the path planning pattern nor traversal of the headland segments in the directions as indicated by the two arrows. (Bottom plot) Concatenation of two pattern elements. The traversal along the "upper" and "lower" headland path is emphasized in blue. Importantly, the area indicated by the red bar can still *not* be reached.

operation of a *fleet* of vehicles, the presented method focuses on the in-field operation of a single vehicle that is repeatedly returning to the field entrance for refilling. This is primarily motivated by the targeted crops (wheat, rapeseed and barley) and the costs of corresponding agricultural vehicles. A support unit, acting as a mobile depot, is assumed to be waiting at the field entrance for refilling. Two comments are made. First, unlike during harvest, mobile units for refilling of spraying tanks cannot come to any arbitrary position along the headland. Second, while not prerequisite for the presented methods, a single field entrance is in line with the objective of repressed area minimization. For aforementioned targeted crops, any new field entrance would result in a new repressed area for the connection of in-field headland path and out-field road network, see Figure 55.



(a) Path planning pattern for CIRC and CIRC*.



(b) Concatenation of two path planning patterns.

Figure 57: (Top plot) Illustration of the second path planning pattern element. The red bar indicates the area that cannot be reached by neither traversal of the path planning pattern element nor traversal of the headland segments in the directions of the two arrows. (Bottom plot) Concatenation of two patterns. Importantly, the area indicated by the red bar in the top plot can now be reached. See the green paths for emphasis of the traversal along the "lower" headland path, and a transition via an interior lane to the "upper" headland path.

3.3.2 Summary

The two path planning patterns under consideration are visualized in Figure 56 and 57. Then, in [145] it is illustrated how the latter pattern is more suitable for partial field coverage. Notice that this pattern also naturally results from methods M1 and M2 from [146] for specific field shapes, see also Figure 46.

3.4 Shortest Path Computations under Trajectory Constraints within Agricultural Fields

This section summarizes [149]:

- M. Graf Plessen, and A. Bemporad, "Shortest path computations under trajectory constraints for ground vehicles within agricultural fields," in *IEEE Conference on Intelligent Transportation Systems*, pp. 1733-1738, 2016.

A method is presented for finding the shortest path to a target point on the boundary of an agricultural working area given a current position and heading of a ground vehicle within that area. Constraints such as admittance of turning on the spot, lane- and corridor-constraining motion, as well as repressed area minimization are taken into account.

3.4.1 Introduction

The agricultural sector is experiencing an increasing degree of automation in both the operation of agricultural machinery, e.g., autonomous guidance of tractors [16], [247], the processing of information [345], and the planning of logistical in-field and inter-field operations [199], [56], [60]. Within this context, this work relates to in-field intelligent transportation systems and logistical optimization. It is applicable both to support a human driver by providing navigation guidance, as well as for the high-level path planning task for a fully autonomous tractor.

We distinguish between two general types of plantable spaces, in the following referred to as *orchard-like areas* and *agricultural fields*. The difference is that for the former, the crop grows on bushes, vines or trees, whereby for the latter the crop is harvested on the ground and, specifically, there also exist headlands equally employed for crop growth. Examples for the former include orchards and vineyards, whereas the latter comprises in particular wheat and rapeseed plantation. For agricultural field operation we further focus on post-seeding operations such as fertilizing and spraying. The distinction between orchard-like areas and

agricultural fields has implications for the operation of machinery, in particular, the importance of minimizing repressed ground area or unimportance thereof.

Shortest path planning is a well known topic in operations research. Popular approaches include dynamic programming (DP) and label correcting algorithms (LCA) such as Dijkstra's method, and A* as a label correcting variation [47].

For existing literature about in-field shortest path planning we refer the reader to [60] and [199]. They treat path planning of service units (SU), e.g., transport wagons, while they are supporting primary units (PU), e.g., harvesters, in the harvesting process. A two-stage optimization method is employed in [60]. In contrast, [199] extends this work to inter-field operation and reduces the optimization to one computational stage. Its focus is on graph modeling with Euclidean distances used as arc costs. For graph search, Dijkstra's method is employed and it is referred to [85] for its implementation. Conceptually, within our work for the navigation in orchard-like areas, the same graph generation approach is taken as in [199]. However, we additionally motivate a heuristic based on a coordinate transformation that decreases the number of LCA-iterations required to find a solution. Additionally, we give details about our graph search algorithms customized to the given problem. The main contribution is the second part about shortest path navigation in agricultural fields, using only existing tractor-lane traces for minimization of repressed ground area. To the best of the authors' knowledge, both the application and solution approach have not been treated before.

3.4.2 Navigation in Orchard-like Areas

Shortest path planning is of relevance in the coordination of SUs and PUs, e.g., in the harvesting process. An alternative application is the navigation in vineyards and orchards. Relevance arises because of limited hectare coverage and the need for a timely return to a depot or mobile station for refilling. For general route planning in orchards, see [55]. See [284] for experiments of an autonomous multi-tractor system that

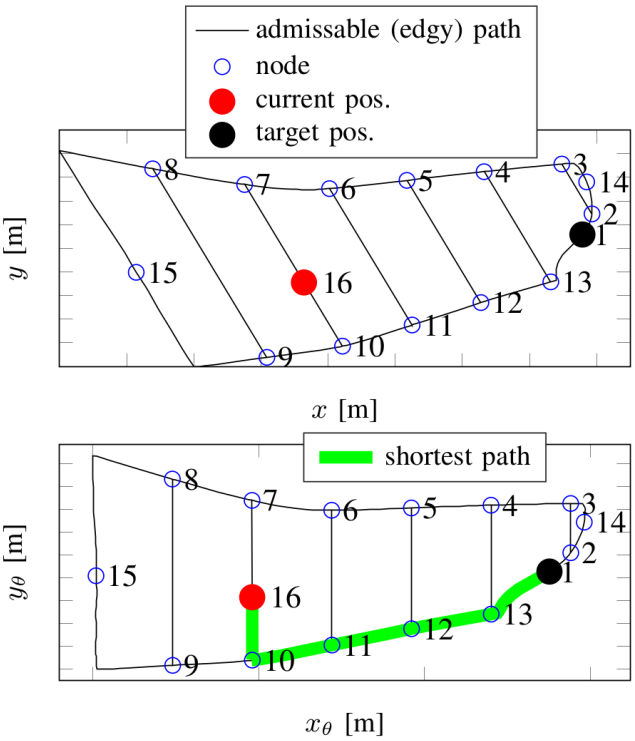


Figure 58: Modeling of the transition graph T with $N = 16$, and illustration of the heuristic from Section 3.4.2 in a rotated coordinate (x_θ, y_θ)-frame.

performs mowing and spraying operations in a citrus orchard. An autonomous navigation system using a 2D laser scanner for straight line recognition of tree rows in an orchard application is presented in [21].

In the following, we distinguish between admitting for the ground vehicle an instantaneous turn on the spot (e.g., plant inspection with a small mobile unit) and vice versa the case of having vehicle motion constrained by small operation corridors or because of larger towed implements prohibiting an instantaneous 180°-turn.

Let

$$\xi_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^\mathsf{T}, \quad i = 1, \ldots, N$$

be the position of *node* $i \in \mathcal{N} = \{1, \ldots, N\}$ in the Universal Transverse Mercator (UTM)-coordinate system with x_i and y_i representing easting and northing-position, respectively. The traverse of an orchard or vineyard can be modeled as a concatenation of transitions between multiple nodes, see Figure 58. We further define *interior lane-segments* (e.g., $13 \rightarrow 4$, $12 \rightarrow 5$, etc.) and *perimetric lane-segments* (e.g., $9 \rightarrow 10$, $10 \rightarrow 11$, etc.). *Auxiliary nodes* (e.g., 15 and 14 in Figure 58) are introduced to ensure unique connections between any two nodes. A transition graph T can then be defined as

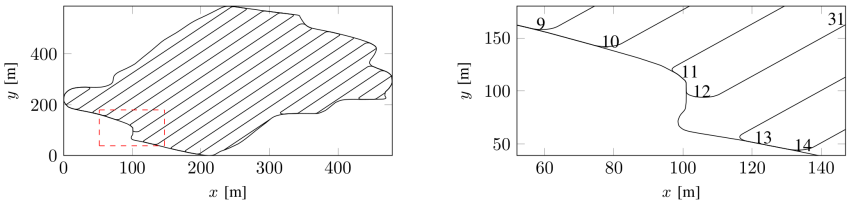
$$T_{ij} = \begin{cases} d_{ij}, & \text{if } \exists \text{ a direct admissible path } i \rightarrow j \\ \infty, & \text{otherwise,} \end{cases} \tag{3.1}$$

where d_{ij} denotes the path length in meters from node i to j and $i, j \in \mathcal{N}$. Throughout, we label our current vehicle position (start) and target position (exit) by node s and e (e.g., nodes 1 and N in Figure 58), respectively. Ultimately, note that with respect to the definition, *interior lane-segments* do not necessarily have to be straights. They may also be curved and thereby shifted in parallel (*freeform*).

In case of *straight* interior lane-segments, let the admissible path-segments and all nodes be described in a new coordinate system defined by $\begin{bmatrix} x_{\theta,i} & y_{\theta,i} \end{bmatrix}^\mathsf{T} = R(\theta) \begin{bmatrix} x_i & y_i \end{bmatrix}^\mathsf{T}$, with $R(\theta)$ being a standard rotation matrix with rotation angle θ. We select θ such that $x_{\theta,m} = x_{\theta,n}$ for all $m, n \in \mathcal{N}$ representing the end-nodes of all interior lane-segments, thereby generating a canonical vertical lane grid, see Figure 58.

Let us define $\Delta x_\theta = |x_{\theta,l} - x_{\theta,m}|$, as the *operating width* of the agricultural machine, where nodes l and m represent *end-points* of two distinct straight interior lane-segments that are both connected via a perimetric lane-segment (i.e., $T_{lm} \neq \infty$). For freeform interior lane-segments, the operating width is defined similarly in a *curvilinear* coordinate system.

For the case of straight interior lane-segments, we abbreviate *"interior lane end-point"* by *ilep*, and further introduce variables $x_\theta^{\max} = \max_{i \in \mathcal{N}} x_{\theta,i}$,



(a) Directed transition curves.

(b) Zoom-in and added numbering.

Figure 59: Illustration of characteristic directed transition curves resulting from natural smoothing by an agricultural machine with limited turning radius.

$x_\theta^{\text{min}} = \min_{i\in\mathcal{N}} x_{\theta,i}$ and

$$x_\theta^{\text{last,max}} = \max_{i\in\mathcal{N}}\{x_{\theta,i} : x_{\theta,i} < \max\{x_{\theta,s}, x_{\theta,e}\},\ x_{\theta,i}\ \textit{ilep}\},$$
$$x_\theta^{\text{last,min}} = \min_{i\in\mathcal{N}}\{x_{\theta,i} : x_{\theta,i} > \min\{x_{\theta,s}, x_{\theta,e}\},\ x_{\theta,i}\ \textit{ilep}\}.$$

Admittance of Turning on the Spot

For determining the shortest path between node s and e, we initialize $J_0(i) = T_{ie}$, $i \in S_0$ with $S_0 = \{i \in \mathcal{N} \ : \ T_{ie} \neq \infty\}$, and define our DP-iteration as

$$J_k(i) = \min_{j\in S_{k-1}} \{T_{ij} + J_{k-1}(j)\},\ i \in S_k, \tag{3.2}$$
$$S_k = \{i \in \mathcal{N} \ : \ T_{ij} \neq \infty,\ j \in S_{k-1}\}, \tag{3.3}$$

which is terminated at a particular stage k on satisfaction of a criterion, that is further discussed below. The shortest path length at DP-iteration stage k from node i to the exit node e is denoted by $J_k(i)$.

An alternative to DP is LCA [47]. Let d_j denote the label of node j, i.e., the path length from starting point s to node j. The general idea of all LCAs is to progressively find shorter paths from the start to every other node j and ultimately the target e. Characteristic is the test

$$d_i + T_{ij} \leq \min\{d_j, \text{UPPER}\}, \tag{3.4}$$

with UPPER $= d_e$. The satisfaction of (3.4) implies that a path from node s to j with i immediately before j is shorter than the current path from s to j and furthermore smaller than the currently shortest path from s to e. Thus, it is a candidate for being part of a shortest path from s to e. In case of satisfaction of (3.4), node j will be further examined at one of the following LCA-iterations for being a candidate for possible inclusion in the shortest path. Besides the sequence of analysis of candidate nodes (breadth-first and depth-first search, Dijkstra's method, etc.), the adaptation of (3.4) by usage of heuristics (to constrain nodes added to the candidate list) is the main driver for reduction in computation times for the discovery of the shortest path.

In the following, we derive a simple heuristic for our application in case of straight interior lane-segments. It can be generalized to freeform interior lanes by transformation to a curvilinear coordinate system.

Proposition 7. *Let the shortest path between current node s and target e be denoted by a list $\mathcal{C} = [c_1, c_2, \ldots, c_l, c_{l+1}, \ldots, c_{l+L-2}, c_{l+L-1}]$ of nodes with $c_1 = s$ and $c_{l+L-1} = e$. Then, it holds that*

$$\begin{aligned} &\textit{if } x_{\theta,s} \leq x_{\theta,e}: \\ &\quad \begin{cases} f^{min} \leq x_{\theta,c} \leq f^{max}, \ \forall c \in \mathcal{C}, \\ x_{\theta,c_{l+1}} \geq x_{\theta,c_l}, \ \forall 1 < l < l+L-2, \end{cases} \qquad (3.5) \\ &\textit{if } x_{\theta,s} > x_{\theta,e}: \\ &\quad \begin{cases} f^{min} \leq x_{\theta,c} \leq f^{max}, \ \forall c \in \mathcal{C}, \\ x_{\theta,c_{l+1}} \leq x_{\theta,c_l}, \ \forall 1 < l < l+L-2 \end{cases} \qquad (3.6) \end{aligned}$$

where $f^{min} = \max\{x_\theta^{last,min} - \Delta x_\theta, x_\theta^{min}\}$ *and* $f^{max} = \min\{x_\theta^{last,max} + \Delta x_\theta, x_\theta^{max}\}$.

Proof. Let us consider the case $x_{\theta,s} \leq x_{\theta,e}$. The proof is analogous for $x_{\theta,s} > x_{\theta,e}$, and is by contradiction. By construction of the transition graph, at every node $i \in \mathcal{N}$ with $\{j \in \mathcal{N} : T_{ij} \neq \infty, \ i \ ilep\}$, there exists at least one j such that $x_{\theta,j} \geq x_{\theta,i}$ and another j such that $x_{\theta,j} \leq x_{\theta,i}$. Thus, a traverse along the nodes connecting interior lane-segments of the transition graph in one of the two monotonous x_θ-directions is always possible. Assume now the shortest path is such that at stage l and node c_l we move to node c_{l+1} with $x_{\theta,c_{l+1}} < x_{\theta,c_l}$. Then, to reach node $x_{\theta,e} > x_{\theta,c_{l+1}}$, the shortest path must pass level x_{θ,c_l} at a stage $l + h > l + 1 > l$

with $h > 1$ because of the connectivity of the graph. By the fact that all nodes $\{i \in \mathcal{N},\ i\ ilep\}$ can be reached by traversing monotonously, we have a contradiction with respect to above assumption about the shortest path. The relation $f^{\min} \leq x_{\theta,c} \leq f^{\max},\ \forall c \in \mathcal{C}$ is then by construction a consequence considering additionally the auxiliary nodes. □

Note that Proposition 7 does not explicitly address the transition directions from and to the starting and exit node, respectively. In general, no a priori rule can be applied since starting and exit nodes may also lie in between two interior lane end-points, and, e.g., in indentations, bays or along highly non-convex perimetric lane-segments.

Proposition 7 further implies that a shortest path from node s to e does consequently not include any turning on the spot, except at most one, at the very beginning at node s.

Proposition 7 can be employed as a heuristic for an efficient implementation of LCA. Thus, any node entering the candidate list must satisfy not only (3.4), but additionally (3.5) in case of $x_{\theta,s} \leq x_{\theta,e}$ and (3.6) in case of $x_{\theta,s} > x_{\theta,e}$.

For DP, iterations (3.2) and (3.3) may be terminated once all nodes with x_θ for which $f^{\min} \leq x_\theta \leq f^{\max}$ holds have been covered. Alternatively, the DP-solutions for traversing from *all* nodes, except s, to the target node e can be stored offline in a look-up table. Then, online the two nodes neighboring the current position s are determined and the corresponding shortest paths starting from both of these neighboring nodes are looked up. The path lengths from the current position are added respectively, and the shorter solution of the two is ultimately selected.

Vehicle Motion Constrained by Operation Corridors

Shortest path planning under consideration of the current heading direction of the ground vehicle, and not permitting any turning maneuvers on the spot due to tight operation corridors, can easily be addressed by constraining the node immediately following after the starting node. Thus, besides $c_1 = s$ and $c_{l+L-1} = e$, we additionally fix c_2 within the shortest path as the node towards the vehicle is invariably heading given the current starting position and orientation.

3.4.3 Navigation in Agricultural Fields

Repressed Area Minimization Constraint

For the coverage of agricultural fields growing, e.g., wheat, rapeseed and similar crops, the transition from an interior lane-segment (lane) to the perimetric tractor-lane (headland path) is in practice *smoothed* by the agricultural machinery, mostly due to its limited turning radius. For autonomous tractor guidance, a corresponding trajectory design approach is presented in [152]. Once a coverage path has been established, see Figure 59 for illustration, it is reasonable to always follow it at every working iteration on that specific field. The reason is repressed area minimization. Consider Figure 59. Suppose instead of driving along the established traces $31 \rightarrow 12 \rightarrow 11$, the ground vehicles covers $31 \rightarrow 12 \rightarrow 13$. Then, at the interior lane-segment $31 \rightarrow 12$ a small new tractor trace would be created, thereby repressing the crop at that location. While it may appear negligible at first sight, it is avoidable and may become very relevant in the sum for all lanes. This is since $L_{\text{loss}} = 2 \cdot l_{\text{repr}} \cdot w_{\text{t}} \cdot g_{\text{gain}}$, where L_{loss} [\$], l_{repr} [m], w_{t} [m] and g_{gain} [\$/m^2] represent total monetary loss due to area repression by tractor traces, the total length of repressing tractor traces, the tire width, and the normalized gain for a particular crop, respectively.

The finding of shortest paths under the *repressed area minimization constraint* may become necessary for the refilling of fertilizers and spraying applications at a farm basis or a mobile depot located outside the agricultural field boundary. In the following, we present a solution approach.

Solution Approach and Algorithm

Let the target node (field exit) be located somewhere along the perimetric lane and the current tractor position anywhere along the tractor traces. We create a transition graph as in (3.1). Additionally, we store all admissible transitions composed of three nodes in a list $\mathcal{U}$. For example, regarding Figure 59, transitions $11 \rightarrow 12 \rightarrow 31$, $14 \rightarrow 13 \rightarrow 12$, etc. are allowed, i.e., $[11, 12, 31] \in \mathcal{U}$. In contrast, $13 \rightarrow 12 \rightarrow 31$ and $31 \rightarrow 12 \rightarrow 13$ are not. We further define $\eta = [s, c_2, e]$, where c_2 represents the node to-

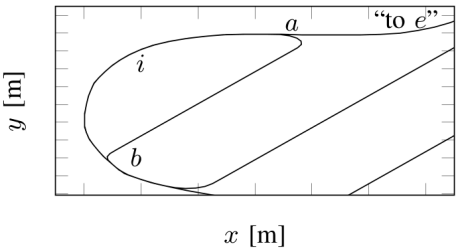


Figure 60: Illustration of the deadlocking principle resulting from the repressed area minimization constraint.

wards which the vehicle is invariably heading given the current starting position and orientation (analogously to the case of constrained vehicle motion due to operation corridors as discussed in Section 3.4.2).

For our purpose we employ a modified LCA with *breadth-first* search, to which we refer as "LCAmod" in the following. Suppose we are currently analyzing transition $i \to j$ for being element of the shortest path from s to e. Then, in the basic LCA implementation test (3.4) is conducted, and, in case of satisfaction, node j is added to the candidate list for further analysis of transitions starting from j. In contrast, in LCAmod, we add j to the candidate list only if (3.4) *and* $[i_{k-1}, i, j] \in \mathcal{U}$ *both* hold, whereby i_{k-1} denotes the current parent-node (predecessor node) of i. We employ LCAmod as outlined in Algorithm 13, where "$-$" denotes one or multiple unused return values of a function call.

Feasibility or existence of a path from s to e can *always* be guaranteed. This is since there is always the solution of transitioning from s to the perimetric tractor-lane and consequently following it until the reaching of e. Noting further that T is created with nonnegative arc costs (here path lengths), it is obvious that there exists a shortest path from s to e. Nevertheless, the single application of LCAmod is not guaranteed to find it. To see this, Figure 60 serves as illustration. The transition from node b to a directly will always be shorter (straight line) than via node i. Thus, the transition $i \to a$ will be discarded when subsequently analyzed as a candidate. As a result, a *deadlock* is reached at node i and no path is determined with segment $i \to a \to$ "to e", eventhough it ob-

viously exists and may even be optimal. To resolve these (and similar) deadlocks, we propose a *restart* of LCAmod. Suppose the node removed last from the candidate list is i with parent-node b. Then, instead of attempting to solve $\eta = [s, c_2, e]$ by one LCAmod-call, we try solving two LCAmod-calls with $\eta^{(1)} = [s, c_2, b]$ and $\eta^{(2)} = [b, i, e]$. A solution to $\eta^{(1)}$ is guaranteed to be found by one LCAmod-call. In contrast, $\eta^{(2)}$ may result in another second deadlock (which we resolve by another restart of LCAmod). Nevertheless, the primary deadlock (transition $i \rightarrow a$) is guaranteed to be overcome. It remains to ensure that the node removed last from the candidate admits together with its parent-node a transition along the perimetric path. (For the illustrating example in Figure 60, node a may also be the node removed last). To ensure this particular transition, we therefore employ the heuristic, that, in case of a deadlock, the node removed last must together with its parent represent a transition along a perimetric lane-segment. Note that more than one restart may be required to find a path from s to e dependent on the complexity of the perimetric contour. Imagine, for example, the case of highly non-convex field shapes with multiple larger bays or indentations. Note that, however, when employing specific heuristic field coverage *patterns* repeatedly on regularly shaped fields, upper bounds on required restarts can be given. Some heuristic field coverage patterns are much more suitable than others. A particularly suitable pattern is displayed in Figure 57. It is discussed in detail in [145].

As outlined above, LCAmod may not immediately in one iteration find a solution to our shortest path problem. We therefore abbreviate the input-output relation of one LCAmod-function call as

$$(\text{UPPER}, \mathcal{C}, d_{\tilde{e}}, i, i_{k-1}) = f(s, c_2, \tilde{e}, T, \mathcal{U}),$$

where i and i_{k-1} refer to the node removed last from the candidate list and its parent when UPPER $= \infty$, respectively. The cost for the shortest path transition from s to $\tilde{e}$ is denoted by $d_{\tilde{e}}$. Usually we have $\tilde{e} = e$. However, as described in Algorithm 13, we may also solve for subproblems with different temporary target nodes. UPPER always refers to the cost to the original target node e.

Algorithm 13: Shortest Paths in Agricultural Fields

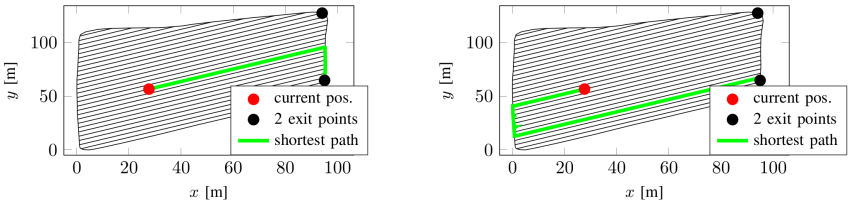
Input: transition graph T, node numbers s, c_2 and e, and all lane-segment coordinates.
for *every node* n *of* N_n *nodes adjacent to* e **do**
    Create a copy T_c of T, cut all connections to e except $n \rightarrow e$ and conduct all following operations on T_c.
    Initialize UPPER $= \infty$, $d_e^{(n)} = 0$, $\mathcal{C}^{(n)} =$ empty list.
    Set $\tilde{s} = s$ and $\tilde{c}_2 = c_2$.
    while *UPPER*$== \infty$ **do**
        $(\text{UPPER}, \mathcal{C}_{e^{(0)}}^{(n)}, d_{e^{(0)}}^{(n)}, i, i_{k-1}) = f(\tilde{s}, \tilde{c}_2, e, T_c, \mathcal{U})$.
        if *UPPER* $== \infty$ **then**
            $(-, \mathcal{C}_{e^{(1)}}, d_{e^{(1)}}, -, -) = f(\tilde{s}, \tilde{c}_2, i_{k-1}, T_c, \mathcal{U})$.
            $(\text{UPPER}, \mathcal{C}_{e^{(2)}}, d_{e^{(2)}}, -) = f(i_{k-1}, i, e, T_c, \mathcal{U})$.
            if *UPPER* $== \infty$ **then**
                Concatenate $\mathcal{C}_{e^{(1)}}$ to $\mathcal{C}^{(n)}$.
                Compute $d_e^{(n)} = d_e^{(n)} + d_{e^{(1)}}$.
                Set $\tilde{s} = i_{k-1}$ and $\tilde{c}_2 = i$.
            else
                Concatenate $\mathcal{C}_{e^{(1)}}$ and $\mathcal{C}_{e^{(2)}}$ to $\mathcal{C}^{(n)}$.
                Compute $d_e^{(n)} = d_e^{(n)} + d_{e^{(1)}} + d_{e^{(2)}}$.
        else
            % Shortest path found in one iteration.
            Set $\mathcal{C}^{(n)} = \mathcal{C}_{e^{(0)}}^{(n)}$ and $d_e^{(n)} = d_{e^{(0)}}^{(n)}$.
Among the N_n solutions, select the one with shortest total path length d_e from s to e, and return the corresponding shortest path-coordinates suitable for navigation-aid or a fully autonomous tractor system application.

Proposition 8. *The shortest path problem within agricultural fields under repressed area minimization constraints can be solved optimally by Algorithm 13.*

Proof. For our application the heading direction is important. We can distinguish between two cases. If a transition is from an interior lane-segment (*interior-to-perimetric*) then there always exists exactly one option to traverse to the perimetric lane. In contrast if a transition is from the perimetric lane (*perimetric-to-interior*), there are either one or two options dependent on the immediately ahead available lane-to-interior tractor trace transition: we can *always* follow the perimetric lane, and *potentially* may turn towards an interior lane-segment.

Any interior lane-segment connects two parts along the perimetric lane. Because of the *breadth-first* search for the removal of nodes from the LCAmod-candidate list for further analysis, both of these parts along the perimetric lane will be analyzed *alternately* after the first available interior-to-perimetric transition. Following above reasoning, this first interior-to-perimetric traversal initiates further a *unique* transition direction, *clockwise* (CW) or *counter-clockwise* (CCW), at two points, referred to as ξ_1 and ξ_2, along the perimetric lane. We introduce the binary variable $\gamma(\xi) \in \{0, 1\}$ to indicate the transition direction as CW and CCW, respectively, initiated at ξ. Thus, because of the aforementioned *perimetric-to-interior* property, it is now possible to *always* follow the perimetric lane starting from both ξ_1 and ξ_2. We can now further distinguish between two scenarios: it may occur $\gamma(\xi_1) == \gamma(\xi_2)$ or $\gamma(\xi_1) \neq \gamma(\xi_2)$. In the former case, then it is easy to see that the target node is guaranteed to be reached without having to restart LCAmod. However, no remark about potential optimality can yet be made. In the latter case, a deadlock *may* result, but does not necessarily have to. In case of a deadlock, it can be resolved by *restarting* LCAmod as discussed above. By Bellman's principle of optimality the shortest path between two general locations A and C can *in general* not be described by the concatenation of, first, the shortest path from A to another location B, and, then, the shortest path from B to C. However, it is the case if a transition from A to C is *not* possible without visiting B. Thus, w.r.t. Algorithm 13, the transition via the deadlock node, achieved by restarting, must be the only possibility to find a path connecting s and e given T_c. This is the case since any other possible transition would have been detected applying the breadth-first search in combination with the perimetric-to-interior transitions.

It remains to be shown that only the iteration over various T_c, with different unique transitions to destination e, yields the shortest path from



(a) Turning on the spot admissable.

(b) Vehicle motion constrained by operation corridors.

Figure 61: Resulting shortest paths for the two experiments with an orchard-like area. (b) The ground vehicle is initially heading along the current interior lane towards negative x- and y-direction. There are two admissible exit points. In both cases (a) and (b), the exit node with smaller y-coordinate resulted in a slightly shorter traveling distance starting from s. The path lengths for the displayed shortest paths are (a) 108m and (b) 171m.

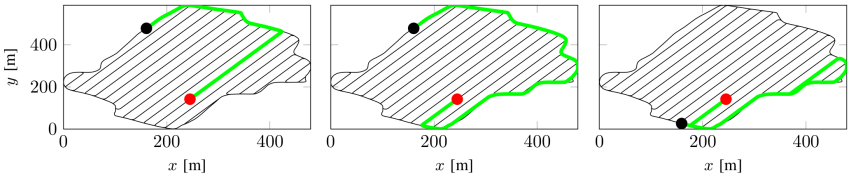
s to e. Suppose we use T directly. Then, a path from s to e may be found. In such a case, all nodes that resulted in a deadlock will have been discarded from the candidate list, eventhough these nodes may have lead to an overall shorter path. This concludes the proof.

□

3.4.4 Numerical Experiments

Orchard-like Areas

We consider a real-world orchard in Northern Germany with an inter-row space of 3.5m (operating width), see Figure 61. For the case of multiple target nodes, we solve the shortest path problem for all of them, before determining the best solution. Algorithmic runtimes are given in Table 6. The LCA-based methods use breadth-first search. We compared breadth-first, depth-first and Dijkstra's graph search method and found breadth-first to be fastest. Eventhough the LCA-heuristic reduces the number of iterations for the given example, it was slightly slower because of the overhead in additional condition checking. All simulations are conducted on a laptop running Ubuntu 14.04 with an Intel Core i7 CPU @2.80GHz×8, 15.6GB of memory, and using MATLAB 8.6 (R2015b).



(a) $\psi_0^{(+)}$, $l_p = 741$m, $N_{\text{restart}} = 0$. (b) $\psi_0^{(-)}$, $l_p = 1189$m, $N_{\text{restart}} = 1$. (c) $\psi_0^{(-)}$, $l_p = 1166$m, $N_{\text{restart}} = 1$.

Figure 62: Resulting shortest paths for experiments within an agricultural field under the repressed area minimization constraint. The initial heading of the ground vehicle along the current interior lane towards positive and negative x- and y-direction is indicated by $\psi_0^{(+)}$ and $\psi_0^{(-)}$, respectively. The total path length and number of restarts required are l_p and N_{restart}, respectively. Three different scenarios are displayed.

Agricultural Fields

Simulation results for a real-world agricultural field in Northern Germany with an operating width (inter-row space) of 24m are displayed in Figure 62. Numerical results are given in Table 7. Interestingly, because of the stringent repressed area minimization constraint check, very few nodes were ultimately added to the candidate list resulting in very fast computation times. For every immediate neighbor of e, a solution is analyzed as outlined in Algorithm 13. The target node for the third experiment in Figure 62(c) has three immediate neighbors. In contrast, for the other two examples in Figure 62(a) and Figure 62(b), there are only two nodes immediately neighboring the exit node. This is the reason for the larger number of 215 required iterations in the third experiment. As indicated in Figure 62(c), despite a local proximity of start and target node, the shortest path between them under the repressed area minimization constraint may be considerably longer and less intuitive.

3.4.5 Conclusion

We presented methods for shortest path finding for ground vehicle operation in agriculturally used areas. It was distinguished between interior

Table 6: Comparison of algorithms for the experiment with an orchard-like area, see Figure 61. The number of required iterations and total computation time [ms] is denoted by N_{iter} and $\bar{\tau}_{\text{total}}$, respectively. Results for both Figures in 61 are given.

		DP	LCA-baseline	LCA-heuristic
Figure 61(a)	$N_{\text{iter}}/\bar{\tau}_{\text{total}}$	10/70.4	44/4.4	20/6.1
Figure 61(b)	$N_{\text{iter}}/\bar{\tau}_{\text{total}}$	11/183.7	54/4.3	20/6.5

Table 7: Comparison of the LCAmod-results for the three experiments displayed in Figure 62. The total number of required LCAmod-iterations and corresponding CPU-time [ms] is denoted by N_{iter} and $\bar{\tau}_{\text{total}}$, respectively.

	Figure 62(a)	Figure 62(b)	Figure 62(c)
$N_{\text{iter}}/\bar{\tau}_{\text{total}}$	54/11.5	114/3.3	215/5.2

lanes and perimetric paths. Constraints such as admittance of an instantaneous turn of the vehicle and, the usage of only already existing tractor traces for repressed area minimization were discussed. Corresponding solution approaches based on dynamic programming and customized label correcting algorithms were given.

3.4.6 Hierarchical Controller Parametrization

The control layer is parameterized by Algorithm 13, which is based on the transition graph T.

3.5 Coupling of Crop Assignment and Vehicle Routing for Harvest Planning in Agriculture

This section is based on [144]:

- M. Graf Plessen, "Coupling of crop assignment and vehicle routing for harvest planning in agriculture," *arXiv preprint arXiv: 1703.08999*, 2017. (Submitted).

A method for harvest planning based on the coupling of crop assignment with vehicle routing is presented. Given multiple fields (up to hundreds), a path network connecting fields, multiple depots at which a number of harvesters are initially located, the main question addressed is: Which crop out of a set of different crops to assign to each field? It must be answered by every farm manager at the beginning of every work-cycle starting with plant seeding and ending with harvesting. Rather than solving a pure assigment problem, we also account for connectivity between fields. In practice, fields are often located distant apart. Traveling costs of machinery and limited harvesting windows demand optimized route planning. The proposed method outputs crop assignment to fields and simultaneously determines an optimized sequence in which to service fields of the same crop during harvest. We derive integer programming (IP) based exact algorithms. For a large numbers of fields where exact algorithms are not tractable anymore, elements of clustering and the solution of local *Traveling Salesman Problems* (TSP) are added that render the method heuristic, but also large-scale applicable.

3.5.1 Introduction

Agriculture is a diverse field ranging from biotech to autonomous robots and finance. At its core, it is related to logistics. According to [2], there are four main functional areas for the agri-food supply chain: production, harvesting, storage and distribution. This work focuses on model-based production planning. In fact, in view of recent plunges of agricultural commodity prices [113], that threaten the sustainability of not few

farmers, efficiency improvements in production are important to minimize unnecessary costs. The decision on the assignment of crops to fields is crucial in that it determines the complete work-cycle. In common practice today, crops are manually clustered according to geographical location and often selected accounting for crop rotation [175] (for reducing soil erosion and increasing soil fertility). The spatial clustering is done for faster harvesting. A trend among farmers in Europe is to collaborate in form of limited companies for sharing of machinery. Not seldomly conflicts arise about the sequence in which to harvest multiple fields of identical crops but various owners. This work is motivated by providing remedy to both the currently as wide-spread and approximate as crucially important practice of crop assignment and the aforementioned conflicts between collaborating farmers by providing a structured methodology.

The *basic multiple Traveling Salesman Problem* (mTSP) describes the objective of finding total tour cost-minimizing routes for m salesmen that all start and end at a single depot, and all vertices are visited once by exactly one salesman, see [31]. Nonnegative edge cost can refer to, e.g., monetary, space or time units. When accounting for various demands at each vertex and limiting the capacity of vehicles (salesmen), the problem is referred to as the capacitated *Vehicle Routing Problem* (VRP). Variations include the VRP with time windows, with backhauls and with pickup and delivery, see [357]. The applications are manifold. For vehicle routing with real-time informations, see for example [219] and the references therein. Recently, there has been increased interest in applying logistical optimization in agriculture for scheduling, routing and fleet management [27], [111], [266], [83], [345]. Special focus was on the *coordination* of machinery teams distinguishing between primary (harvester) and service (transport) units referred to as PUs and SUs, see [56], [199], [335], [299]. All of these references assume that fields with assigned crops are given. To the author's knowledge, the optimized assignment of crops to fields and *simultaneously* accounting for vehicle routing and other constraints for optimized harvest planning has not been discussed in the literature. We propose such strategic assignment to be conducted once at the *beginning* of the work-cycle, thereby decisively affecting the complete

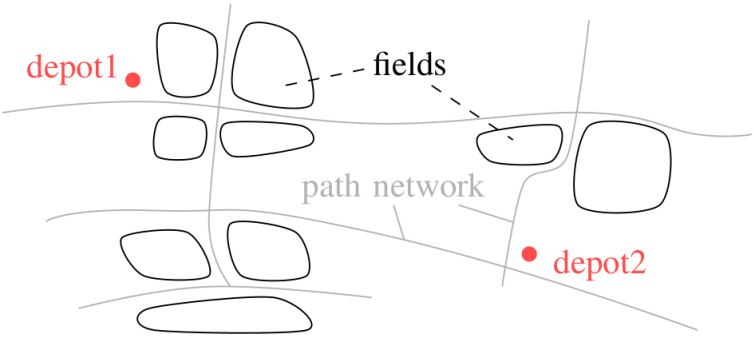


Figure 63: Visualization of three key components for planning: multiple fields, a path network connecting the fields, and multiple depots. At each depot, multiple harvesters may initially be based (fourth key component).

agricultural production-cycle. The second step involves coordinations of PUs and SUs, exploiting all of the aforementioned references, and is to be conducted at the *end* of the work-cylce during harvest.

3.5.2 Problem Formulation and Notation

Problem Formulation

For optimized harvest planning in agriculture, we consider four key infrastructural components illustrated in Figure 63. See also Figure 64 for problem visualization. At the beginning of every seasonal work-cycle, a crop has to be assigned to all available fields, and the following four interrelated questions must be addressed:

1. Which *crop* to optimally assign to each field?
2. In what *sequence* to optimally service all fields during harvest?
3. How to optimally dispatch *multiple harvesters* that initially may be located at multiple depots to the multiple fields?
4. Which fields should be serviced, and which *leased* instead, and at what prices?

Figure 64: Problem visualization. Yellow markers indicate fields to be served by collaborating farms. Overall, there are 85 fields. The satellite image shows an area of 15.9 $\times$ 16.3km. The path network connecting the fields is curvy and often along rural gravel roads only permitting slow traveling speeds. The overall field coverage area is more than 1700ha. Traveling distances between pairs of fields is between meters up to dozens of km.

The first question decides the complete seasonal work-cycle of the farm. For optimized harvest planning, its answer must *simultaneously* account for questions 2) to 4).

Optimization problems are derived that permit to input parameters such as, e.g., yields per field and crop. The selection of these parameters largely determines the output of the optimization, and should be based on a *modeling* step involving historical data (experience). Then, at the *end* of the work-cycle at harvest, deviations from initial modeling typically have occurred. For example, the actual amount of crop harvested

per field is different from the predicted one, and weather is influencing potential harvesting-windows. Thus, at the *end* of every work-cycle, the aforementioned second step becomes relevant, which then involves the coordination of PUs and SUs [143]. Below, we focus on planning at the *beginning* of the (yearly) work-cycle[1].

Notation

Let us introduce notation mainly adopting [357]. We denote a complete graph $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{0, \dots, D-1, D, \dots, D+L-1\}$ and $\mathcal{A}$ are vertex and arc set, respectively. The cardinality of a set of vertices is denoted by $|\cdot|$. Vertices $i \in \mathcal{D} = \{0, \dots, D-1\}$ and $i \in \mathcal{L} = \{D, \dots, D+L-1\}$ correspond to D depots and L fields. The corresponding geographical coordinates are denoted by $P_l = (X_l, Y_l) \in \mathbb{R}^2,\ \forall l \in \mathcal{L}$, and similarly $P_d,\ \forall d \in \mathcal{D}$. The K difference crops are indexed by $\mathcal{K} = \{0, \dots, K-1\}$. Let the number of harvesters located at a depot and suitable for a crop be denoted by $N_d^{\text{harv},k},\ \forall d \in \mathcal{D},\ \forall k \in \mathcal{K}$. Let the normalized traveling cost per harvester and crop k between a depot d and a field j or between two fields i and j be denoted by $\tilde{c}_{dj}^k$ and $\tilde{c}_{ij}^k$, respectively. Abbreviating $N^{\text{harv},k} = \sum_{d \in \mathcal{D}} N_d^{\text{harv},k}$, we define traveling costs as follows:

$$c_{ij}^k = N^{\text{harv},k} \tilde{c}_{ij}^k,\ \forall i, j \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.7}$$

$$c_{dj}^k = N^{\text{harv},k} \tilde{c}_{dj}^k,\ \forall d \in \mathcal{D},\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.8}$$

$$c_{dj}^{k,k_{\min}} = \sum_{\tilde{d} \in \mathcal{D}} N_{\tilde{d}}^{\text{harv},k} \tilde{c}_{\tilde{d}j}^k,\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}. \tag{3.9}$$

Furthermore, we define c_{jd}^k and $c_{jd}^{k,k_{\max}}$ similarly to (3.8) and (3.9), respectively. Note that traveling costs along the same geographical paths may vary for different k due to different crop-dependent operating machinery. The (expected) revenue from growing and marketing of crop $k \in \mathcal{K}$ on field $l \in \mathcal{L}$ is denoted by r_l^k. We assume a fixed cost of γ is incurred for every additional crop. Maintenance cost per depot are given by $n^d,\ \forall d \in \mathcal{D}$. All costs shall be in monetary units.

[1] In general, different crops may have different (not necessarily *yearly*) work-cycle durations. In Northern Germany and for the crops considered (barley, rapeseed and wheat), the typical work-cycle duration (from preparation to harvesting) is assumed as one year.

Let us discuss decision variables. We distinguish between two major classes: *natural* and *auxiliary* decision variables. The first class comprises binary $x_{ij}^k \in \{0,1\}, \forall i,j \in \mathcal{V}, \forall k \in \mathcal{K}$ with $x_{ij}^k = 1$ indicating arc (i,j) to be element of the optimal route for crop k. Symmetries are exploited whenever possible, i.e., x_{ji}^k is dismissed whenever $x_{ji}^k = x_{ij}^k$. For the symmetric case, we also assign $x_{dj}^k \in \{0,1,2\}, \forall d \in \mathcal{D}, \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}$, thereby indicating a visit of only field j for route corresponding to crop k. Further, there are binary $\delta_l^k \in \{0,1\}, \forall l \in \mathcal{L}, \forall \mathcal{K}$, with $\delta_l^k = 1$ indicating that crop k is assigned to field l. Integer m is such that $1 \leq m \leq K$ indicates the number of active crops in the optimal solution. As will be shown, *auxiliary* decision variables result from incorporating *logical* constraints into integer problems.

3.5.3 Problem approach

Framework and Approach

Assumption 2. *Different crops have different non-overlapping harvesting times.*

Assumption 3. *Throughout the harvest of any crop, harvesters are usually refueled and maintained on fields (i.e., there is no intermediate return to depots).*

Based on Assumptions 2 and 3, we approach above problem formulation using a mTSP-framework [31]. A route for each crop (crop-tour) and the fields correspond to a traveling salesman route and cities to be visited, respectively. Eventhough m routes (one for m crops) are planned simultaneously, the *sequential* harvesting times ultimately permit the framework. The mTSP-problem requires modifications to combine with crop assignments and to account for more specialized constraints. To name just one example, a state space extension is required, i.e., adding the crop-dimension k to obtain x_{ij}^k instead of x_{ij} used in basic mTSP-formulations. We employ an integer programming (IP) framework for its ability to incorporate various constraints.

Clustering

A useful tool for us is grouping or *clustering* of fields, e.g., via the $\tilde{k}$-means algorithm [173]. As will be discussed below, it enables to upscale

the number of fields that can be handled in a structured manner coupling crop assignment and route planning.

Pure Assignment Problems

The most basic IP for pure crop assignment to fields (without accounting for routing) is:

$$\min \quad -\sum_{l\in\mathcal{L}}\sum_{k\in\mathcal{K}} r_l^k \delta_l^k \tag{3.10a}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}} \delta_l^k = 1,\ \forall l \in \mathcal{L}, \tag{3.10b}$$

$$\delta_l^k \in \{0,1\},\ \forall l \in \mathcal{L},\ \forall k \in \mathcal{K}. \tag{3.10c}$$

Under additional assumptions $r_l^k = r^k,\ \forall l$ and $r^k \neq r^j,\ \forall k \neq j$, its optimal solution always assigns the most profitable crop (with largest r^k) to *all* fields. Let us discuss types of constraints that can be added.

First, we mention hard equality constraints motivated, for example, by *crop rotation* [175] or soil considerations (specific soils only admit specific crops),

$$\delta_l^k = 0,\ \forall (l,k) \in \mathcal{R}, \tag{3.11}$$

where $\mathcal{R}$ denotes a set of prohibited field/crop-combinations. Throughout, we assume that $\mathcal{R} = \{(l,k) : \sum_{k\in\mathcal{K}} \delta_l^k > 0,\ \forall l \in \mathcal{L}\}$, i.e., for every field there is at least one crop always admissible.

Second, we mention *diversification* inequality constraints

$$\sum_{l\in\mathcal{L}} g_l^k \delta_l^k \leq G^k,\ \forall k = 0,\ldots,K-2, \tag{3.12}$$

with $g_l^k \geq 0$ denoting weights (for example the hectares-coverage or required production means for field l and crop k) and $G^k \geq 0$ the corresponding crop-related bounds, thereby *diversifying* crop-growth. Note that one crop $k = K-1$ was left unconstrained for feasibility. In general, when combining *both* aforementioned hard and inequality constraints without additional precaution, feasibility of the resulting IP cannot be guaranteed. Infeasibility results if these constraints enforce $\sum_{k\in\mathcal{K}} \delta_l^k = 0$, thereby violating (3.10b).

When including both crop rotation and diversification constraints, replacing (3.10a) by the relaxation $\sum_{k\in\mathcal{K}} \delta_l^k \leq 1$ always guarantees feasibility of (3.10). This is since these constraints can always be satisfied by $\delta_l^k = 0$.

Proposition 9. *The solution of the LP-relaxation of IP* (3.10)*, and also including crop rotation constraints* (3.11)*, is integer feasible, and thus solves these problems as well.*

Proof. We can easily summarize the IP as $\min\{c^\mathsf{T}x : Ax = b,\ x_l \in \{0,1\}, \forall l = 0,\dots,KL-1\}$. Its LP-relaxation reads $\min\{c^\mathsf{T}x : Ax = b,\ x \geq 0\}$. By [332], if $\tilde{A}$ is *totally unimodular*, the LP $\min\{\tilde{c}^\mathsf{T}\tilde{x} : \tilde{A}\tilde{x} = \tilde{b},\ \tilde{x} \in \mathbb{R}_+^n\}$ has an integral optimal solution for all integer vectors $\tilde{b}$ for which it has a finite optimal value. It thus remains to show that A associated with the LP-relaxation of our IP is totally unimodular. By [180], a matrix A is totally unimodular if: (i) each entry is 0, 1 or -1; (ii) each column contains at most two non-zeros; (iii) the set $\mathcal{N}$ or row indices of A can be partitioned into $\mathcal{N}_1 \cup \mathcal{N}_2$ such that in each column l with two non-zeros we have $\sum_{m_1\in\mathcal{N}_1} a_{m_1 l} = \sum_{m_2\in\mathcal{N}_2} a_{m_2 l}$. Condition (i) is trivially true from (3.10b) and (3.11). Regarding (ii), (3.10b) implies exactly one nonzero coefficient equal to 1 per column; to which, by (3.11), at most one more nonzero coefficient equal to 1 is added. For (iii), we partition sets $\mathcal{N}_1$ and $\mathcal{N}_2$ according to constraints (3.10b) and (3.11). Then $\sum_{m_1\in\mathcal{N}_1} a_{m_1 l} = 1$ and $\sum_{m_2\in\mathcal{N}_2} a_{m_2 l} = 1$ using the previous argument for (ii). This concludes the proof. □

The consequence of Proposition 9 is that very large instances (with many fields and crops) of (3.10) with (3.11) can easily be solved. This is since there exist very efficient linear programming solvers. As a remark, the aforementioned inequality relaxation of (3.11) does not affect the totally unimodular property. This is since slack variables s_l can be introduced such that $\sum_{k\in\mathcal{K}} \delta_l^k + s_l = 1,\ s_l \in \{0,1\},\ \forall l \in \mathcal{L}$. They are not affecting (3.12), and thus a similar corresponding proposition and proof can be formulated. In contrast, adding diversification constraints (3.12), in general, render the LP-relaxation to not be integer feasible anymore.

TSP with Different Start and End Node

A useful tool for us is the TSP with different start and end node. As will be outlined below, it is employed for the routing within clusters of fields planting the same crop. Therefore, dropping superscript k, the IP is:

$$\min \quad \sum_{i<j} c_{ij} x_{ij} \tag{3.13a}$$

$$\text{s.t.} \quad \sum_{j=1}^{N-2} x_{0j} = 1, \ \sum_{j=1}^{N-2} x_{jN-1} = 1, \tag{3.13b}$$

$$\sum_{i<j} x_{il} + \sum_{l<j} x_{lj} = 2, \ l = 1, \dots, N-2, \tag{3.13c}$$

$$\sum_{i<j; i,j \in S} x_{ij} \leq |S| - 1, \ 3 \leq |S| \leq N-3, \quad \forall S \subseteq \mathcal{V} \backslash \{0, N-1\}, \tag{3.13d}$$

$$x_{ij} \in \{0,1\}, \ 0 \leq i < j, \ j = 1, \dots, N-1, \tag{3.13e}$$

whereby we here set node 0 and $N-1$ as start and end node among a cluster of N nodes. Constraints (3.13b) and (3.13c) indicate that start and end node are incident to one edge, and all other nodes incident to two, respectively. Under the assumption of symmetric edge weights, the *subtour elimination constraints* (SECs) [239] are given by (3.13d).

A Remark to Incorporating SECs in Integer Programs

Formulation (3.13) as well as the IPs following in Section 3.5.4 include an exponential number of SECs [239]. We approach SECs in form of *separation algorithms* [303], i.e., by adding SECs sequentially as they are needed. With regard of (3.13), we start by solving it without (3.13d). If the result does not return any subtour, we have found the optimal solution. Otherwise, all detected subtours are added to (3.13) as SECs, and the IP is solved again. This is repeated until a solution without subtours is found (the optimal solution), or a maximal number of SEC-iterations is reached.

Harvesters Traveling as a Group

Remark 7. *Assume that all of multiple harvesters are initially located at one depot to which they must return after processing all fields associated with a crop. An optimal policy is that all harvesters cover the fields together as a group, i.e., without distributing harvesters among different fields of the same crop.*

Proof. Harvesters are not constrained by each other. They can always work in parallel on each field. The asymmetric case with fields ripening at different times already implies a unique optimal working sequence. For the symmetric case, an optimal route includes exactly two edges incident to the depot vertex. By symmetry it is thus always an optimal solution that all harvesters travel as a group. Any other initial distribution of harvesters to fields not connected to the depot vertex along the two aforementioned edges is already suboptimal by nonnegativity of traveling costs. □

The concept of multiple harvesters traveling as a group according to Remark 7 bears more advantages. In general, SUs must ideally be operated such that PUs (harvesters) can operate continuously to avoid any waiting times. The rate at which harvesters are filled is not perfectly predictable due to varying crop returns, even within fields. Having concentrated all SUs to one field is beneficial for robustness in that *multiple* harvesters can be served (instead of specific SU-PU couples) according to short-term freed capacities. Another advantage is the facilitated supervision by the farm-manager.

Remark 7 has further implications for the general setting in which multiple harvesters are initially distributed at multiple depots. Optimization can be conducted *depot*-wise (instead of single harvester-wise). Besides this, no general a priori remark about routing of harvester groups (depot groups) can be made. One heuristic is that all harvesters assemble at the first field of a crop-route and then proceed group-wise. Such method is motivated by the fact that a timely agreement on harvest-start, e.g., a day ahead, permit all depot-groups to plan the travel in time and consequently start field and route coverage coordinatedly. In general, though, a distributed assignment of depot groups to fields is optimal. Consider depots distant apart with fields clustered around each depot.

Logical Constraints

The integration of logical constraint in optimization problems is discussed in Section 1.3. Three classes of logical constraints are of particular interest. As indicated, these can be translated into integer linear inequalities.

Priority Constraints

To account for *a priori* experience about different sequences in ripeness of fields, *priority constraints* can be formulated. For example, relating to uncertainties, the sequence in which fields of the same crop ripe may vary, e.g., due to hillsides and varying soil. W.l.o.g., consider a statement such as "if fields a, b, and c are among the ones assigned to crop k, then the corresponding sequence for harvest shall be in order c, a and b". This can be modeled as $x^k_{ca} = \delta^k_c \delta^k_a$ and $x^k_{ab} = \delta^k_a \delta^k_b$ and can be translated to *linear* integer inequalities by means of (1.10). Thus a sequential node-by-node procedure (first vertices (c, a), then (a, b), etc.) is recommended. Note that an *asymmetric* edge model has to be employed for all connections between vertices for which priorities are defined. For above example, we require $x^k_{ca} \neq x^k_{ac}$. Otherwise, there is no *direction* information.

3.5.4 Problem Solution

Integer Linear Programming

We propose eight integer linear programs, denoted by IP-n, $n = 1, \ldots, 8$. IP-n and IP-$(n + 4)$ for $n = 1, \ldots, 4$ are identical except that for the latter four, all K crops are enforced to be included in the solution, whereas the former four are formulated to also permit only any subset of K crops. This distinction has significant influence on computational complexity and problem formulation as will be shown. Throughout this section, we use indices according to $d \in \mathcal{D}$, $i, j, l \in \mathcal{L}$ and $k \in \mathcal{K}$. Because of Assumption 2, we order crops in $\mathcal{K}$ such that a low index indicating an earlier harvesting time. Throughout, for the procedure of solving IPs, the SECs are handled as outlined in Section 3.5.3.

IP-1. Let there be one depot $d \in \mathcal{D}$ from which all harvesters start and to which all harvesters return after each crop-route. According to Remark 7, all harvesters are dispatched as a group. We therefore propose the following IP:

$$\begin{aligned}
\min \ & \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} c_{dj}^k x_{dj}^k + \sum_{k\in\mathcal{K}}\sum_{i<j} c_{ij}^k x_{ij}^k - \sum_{l\in\mathcal{L}}\sum_{k\in\mathcal{K}} r_l^k \delta_l^k \\
& + \gamma m + n^d \eta^d && \text{(3.14a)} \\
\text{s.t. } & x_{dl}^k + \sum_{i<l} x_{il}^k + \sum_{l<j} x_{lj}^k = 2\delta_l^k, \ \forall l \in \mathcal{L}, \ \forall k \in \mathcal{K}, && \text{(3.14b)} \\
& \sum_{k\in\mathcal{K}} \delta_l^k = 1, \ \forall l \in \mathcal{L}, && \text{(3.14c)} \\
& \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} x_{dj}^k = 2m, && \text{(3.14d)} \\
& \sum_{i<j;i,j\in S^k} x_{ij}^k \le |S^k| - 1, \ 3 \le |S^k| \le N-1, \\
& \qquad \forall k \in \mathcal{K}, \ S^k \subseteq \mathcal{V}\backslash\{d\}, && \text{(3.14e)} \\
& x_{dj}^k \in \{0,1,2\}, \ \forall j \in \mathcal{L}, \ \forall k \in \mathcal{K}, && \text{(3.14f)} \\
& x_{ij}^k \in \{0,1\}, \ 0 \le i < j, \ \forall k \in \mathcal{K}, && \text{(3.14g)} \\
& \delta_l^k \in \{0,1\}, \ \forall l \in \mathcal{L}, \ \forall k \in \mathcal{K}, && \text{(3.14h)} \\
& 1 \le m \le K, && \text{(3.14i)}
\end{aligned}$$

with decision variables x_{dj}^k, x_{ij}^k, δ_l^k, m and assuming symmetric edge costs. In case of asymmetry regarding the traversal to and from depot vertices, we can replace $\sum_{d\in\mathcal{D}} x_{dl}^k$ and $x_{dl}^k \in \{0,1,2\}$ with $\sum_{d\in\mathcal{D}} x_{dl}^k + \sum_{d\in\mathcal{D}} x_{ld}^k$ and $x_{dl}^k, x_{ld}^k \in \{0,1\}$, and similarly adopt (3.14d). *Diversification*, *crop rotation* and *priority* constraints discussed in Section 3.5.3 can easily be added. Similarly, constraints on the total traveling cost per crop can be formulated. For example, for constraints on *traveling time* we can formulate

$$\sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}} h_{dj}^k x_{dj}^k + \sum_{i<j} h_{ij}^k x_{ij}^k \le T_{\text{win}}^k - \sum_{l\in\mathcal{L}} T_l^{\text{harv},k} \delta_l^k, \ \forall k \in \mathcal{K} \qquad \text{(3.15)}$$

where, for generality, we assumed the multi-depot case, and where h_{dj}^k

and h_{ij}^k denote travel time along corresponding edges, T_{win}^k the harvesting window for crop k (typically multiple days), and $T_l^{\text{harv},k}$ the required harvesting time (typically proportional to number of active harvesters) per field l and crop k. For a large number of fields, (3.15) become crucial because of limited harvesting time windows. In fact, in combination with the *partial service* constraints outlined in Section 3.5.5 they are central to limiting the maximum number of fields that should be serviced to still add monetary value. Note that without additional precaution *or* the relaxed service constraints to be discussed in Section 3.5.5, (3.15) may invoke infeasible IPs. IP (3.14) has $n_z = KL + K\sum_{q=0}^{L-2} L - 1 - q + KL + 1$ integer decision variables. Term $n^d\eta^d$ in (3.14a) is constant since one depot is considered only. Constraints (3.14b) in combination with (3.14c) indicate that every field l is assigned exactly one crop k, and every field is incident to exactly two edges. Thus, these type of constraints couple crop assignment with vehicle routing. Constraint (3.14d) enforces the depot node to have exactly m edges incident, where m is a decision variable according to (3.14i). Under the assumption of symmetric edge weights, the SECs are given by (3.14e). By construction, any subtour is associated with exactly one crop $k \in \mathcal{K}$.

IP-2. Harvesters start the first crop-route from multiple depots. After *each* crop-route they must return to their original start depots. For IP-2, and under the assumption of symmetric traveling costs, we model $c_{dj}^k = c_{dj}^{k,k_{\min}},\ \forall k \in \mathcal{K}$. The remainder of IP-2 is identical to (3.14).

IP-3. Among a group of multiple available depots we seek the optimal selection. We assume that after every crop-route all harvesters will consequently start from and return to the optimal depot selection. The following IP is therefore proposed:

$$
\begin{aligned}
\min \quad & \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} c_{dj}^k x_{dj}^k + \sum_{k\in\mathcal{K}}\sum_{i<j} c_{ij}^k x_{ij}^k \\
& - \sum_{l\in\mathcal{L}}\sum_{k\in\mathcal{K}} r_l^k \delta_l^k + \gamma m + \sum_{d\in\mathcal{D}} n^d \eta^d && \text{(3.16a)} \\
\text{s.t.} \quad & \sum_{d\in\mathcal{D}} x_{dl}^k + \sum_{i<l} x_{il}^k + \sum_{l<j} x_{lj}^k = 2\delta_l^k,\ \forall l\in\mathcal{L},\ \forall k\in\mathcal{K}, && \text{(3.16b)} \\
& \sum_{k\in\mathcal{K}} \delta_l^k = 1,\ \forall l\in\mathcal{L}, && \text{(3.16c)} \\
& \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} x_{dj}^k = 2p^d,\ \forall d\in\mathcal{D}, && \text{(3.16d)} \\
& \sum_{d\in\mathcal{D}} \eta^d = 1, && \text{(3.16e)} \\
& \eta^d \le p^d \le K\eta^d,\ \forall d\in\mathcal{D}, && \text{(3.16f)} \\
& p^d \le m - (1-\eta^d),\ \forall d\in\mathcal{D}, && \text{(3.16g)} \\
& p^d \ge m - K(1-\eta^d),\ \forall d\in\mathcal{D}, && \text{(3.16h)} \\
& \sum_{i<j;i,j\in S^k} x_{ij}^k \le |S^k| - 1,\ 3\le|S^k|\le N-1, \\
& \qquad\qquad \forall k\in\mathcal{K},\ S^k\subseteq\mathcal{V}\backslash\{d\},\ \forall d\in\mathcal{D}, && \text{(3.16i)} \\
& x_{dj}^k \in \{0,1,2\},\ \forall d\in\mathcal{D},\ \forall j\in\mathcal{L},\ \forall k\in\mathcal{K}, && \text{(3.16j)} \\
& x_{ij}^k \in \{0,1\},\ 0\le i<j,\ \forall k\in\mathcal{K}, && \text{(3.16k)} \\
& \delta_l^k \in \{0,1\},\ \forall l\in\mathcal{L},\ \forall k\in\mathcal{K}, && \text{(3.16l)} \\
& 1\le m\le K, && \text{(3.16m)} \\
& \eta^d \in \{0,1\},\ \forall d\in\mathcal{D}, && \text{(3.16n)} \\
& p^d \in \{0,1,\ldots,K\},\ \forall d\in\mathcal{D}, && \text{(3.16o)}
\end{aligned}
$$

with decision variables $x_{dj}^k,\ x_{ij}^k,\ \delta_l^k,\ m,\ \eta^d$ and p^d. IP (3.16) has $N_z = KL + K\sum_{q=0}^{L-2} L - 1 - q + KL + 1 + 2D$ integer decision variables. We discuss the key distinction w.r.t. IP-1. Since $D > 1$ is assumed, we model

the decision to start from one of the depots as equality constraints

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} x_{dj}^k = 2m\eta^d,\ \forall d\in\mathcal{D}, \tag{3.17}$$

with $x_{dj}^k \in \{0,1,2\}$, $\eta^d \in \{0,1\}$, $1 \le m \le K$ and $\sum_{d\in\mathcal{D}}\eta^d = 1$. Since (3.17) is nonlinear, we introduce auxiliary variable $p^d = m\eta^d,\ \forall d\in\mathcal{D}$. By (1.11), this can be translated to linear inequality constraints (3.16f), (3.16g) and (3.16h). The cost coefficients c_{dj}^k are according to (3.8).

IP-4. Harvesters start the first crop-route from multiple depots. Then, they assemble at the first field for that route and consequently travel as one group until the last field to be covered for the last crop-route. With the exception of the last crop-route, all harvesters return to one depot which is selected optimally among the group of available depots. After the last field of the last crop-tour, all harvesters return to their original start depots. We therefore propose:

$$\begin{aligned}
\min\ & \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} c_{dj}^{k,k^{\min}} v_{dj}^k + c_{dj}^k x_{dj}^k - c_{dj}^k v_{dj}^k + \\
& \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} c_{jd}^{k,k^{\max}} w_{jd}^k + c_{jd}^k x_{jd}^k - c_{jd}^k w_{jd}^k \\
& \sum_{k\in\mathcal{K}}\sum_{i<j} c_{ij}^k x_{ij}^k - \sum_{l\in\mathcal{L}}\sum_{k\in\mathcal{K}} r_l^k\delta_l^k + \gamma m + \sum_{d\in\mathcal{D}} n^d\eta^d && \text{(3.18a)}\\
\text{s.t.}\ & \sum_{d\in\mathcal{D}} x_{dl}^k + \sum_{i<l} x_{il}^k + \sum_{l<j} x_{lj}^k + \sum_{d\in\mathcal{D}} x_{ld}^k = 2\delta_l^k, \begin{cases}\forall l\in\mathcal{L},\\ \forall k\in\mathcal{K},\end{cases} && \text{(3.18b)}\\
& \sum_{k\in\mathcal{K}} \delta_l^k = 1,\ \forall l\in\mathcal{L}, && \text{(3.18c)}\\
& \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} x_{dj}^k + x_{jd}^k = 2p^d,\ \forall d\in\mathcal{D}, && \text{(3.18d)}\\
& \sum_{d\in\mathcal{D}} \eta^d = 1,\ \sum_{k\in\mathcal{K}} \tilde{a}^k = 1,\ \sum_{k\in\mathcal{K}} \tilde{\beta}^k = 1, && \text{(3.18e)}\\
& \tilde{\alpha}^0 = \alpha^0,\ \tilde{\beta}^{K-1} = \alpha^{K-1}, && \text{(3.18f)}\\
& \sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}}\sum_{k\in\mathcal{K}} v_{dj}^k = 1,\ \sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}}\sum_{k\in\mathcal{K}} w_{jd}^k = 1, && \text{(3.18g)}
\end{aligned}$$

and continued

$$\sum_{j\in\mathcal{L}} x_{dj}^k = \eta^d,\ \forall d\in\mathcal{D},\ \forall k\in\mathcal{K}, \tag{3.19a}$$

$$\sum_{j\in\mathcal{L}} x_{jd}^k = \eta^d,\ \forall d\in\mathcal{D},\ \forall k\in\mathcal{K}, \tag{3.19b}$$

$$\eta^d \le p^d \le K,\ \forall d\in\mathcal{D}, \tag{3.19c}$$

$$p^d \le m - (1-\eta^d),\ \forall d\in\mathcal{D}, \tag{3.19d}$$

$$p^d \ge m - K(1-\eta^d),\ \forall d\in\mathcal{D}, \tag{3.19e}$$

$$1 - \sum_{l\in\mathcal{L}} \delta_l^k \le 1-\alpha^k,\ \forall k\in\mathcal{K}, \tag{3.19f}$$

$$1 - \sum_{l\in\mathcal{L}} \delta_l^k \ge \epsilon + (-L+1-\epsilon)\alpha^k,\ \forall k\in\mathcal{K}, \tag{3.19g}$$

$$\alpha^k + (1 - \sum_{\tau=0}^{k-1}\tilde{\alpha}^\tau) - \tilde{\alpha}^k \le 1,\ \forall k = 1,\dots,K-1, \tag{3.19h}$$

$$\tilde{\alpha}^k \le \alpha^k,\ \tilde{\alpha}^k \le 1 - \sum_{\tau=0}^{k-1}\tilde{\alpha}^\tau,\ \forall k = 1,\dots,K-1. \tag{3.19i}$$

$$\alpha^{K-2-k} + (1 - \sum_{\tau=1}^{1+k}\tilde{\beta}^{K-2-k+\tau}) - \tilde{\beta}^{K-2-k} \le 1, \quad \forall k = 0,\dots,K-2, \tag{3.19j}$$

$$\tilde{\beta}^{K-2-k} \le \alpha^{K-2-k},\ \forall k = 0,\dots,K-2, \tag{3.19k}$$

$$\tilde{\beta}^{K-2-k} \le 1 - \sum_{\tau=1}^{1+k}\tilde{\beta}^{K-2-k+\tau},\ \forall k = 0,\dots,K-2, \tag{3.19l}$$

$$\tilde{\alpha}^k + x_{dj}^k - v_{dj}^k \le 1,\ v_{dj}^k \le \tilde{\alpha}^k,\ v_{dj}^k \le x_{dj}^k, \quad \forall d\in\mathcal{D},\ \forall j\in\mathcal{L},\ \forall k\in\mathcal{K}, \tag{3.19m}$$

$$\tilde{\beta}^k + x_{jd}^k - w_{jd}^k \le 1,\ w_{jd}^k \le \tilde{\beta}^k,\ w_{jd}^k \le x_{jd}^k, \quad \forall d\in\mathcal{D},\ \forall j\in\mathcal{L},\ \forall k\in\mathcal{K}, \tag{3.19n}$$

$$\sum_{i<j;i,j\in S^k} x_{ij}^k \le |S^k| - 1,\ 3 \le |S^k| \le N-1, \quad \forall k\in\mathcal{K},\ S^k \subseteq \mathcal{V}\backslash\{d\}, \tag{3.19o}$$

with decision variables

$$x_{dj}^k \in \{0,1\},\ \forall d \in \mathcal{D},\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.20a}$$

$$x_{ij}^k \in \{0,1\},\ 0 \leq i < j,\ \forall k \in \mathcal{K}, \tag{3.20b}$$

$$\delta_l^k \in \{0,1\},\ \forall l \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.20c}$$

$$1 \leq m \leq K, \tag{3.20d}$$

$$\eta^d \in \{0,1\},\ \forall d \in \mathcal{D}, \tag{3.20e}$$

$$p^d \in \{0,1,\dots,K\},\ \forall d \in \mathcal{D}, \tag{3.20f}$$

$$\alpha^k, \tilde{\alpha}^k, \tilde{\beta}^k \in \{0,1\},\ \forall k \in \mathcal{K}, \tag{3.20g}$$

$$v_{dj}^k, w_{jd}^k \in \{0,1\},\ \forall d \in \mathcal{D},\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.20h}$$

$$x_{jd}^k \in \{0,1\},\ \forall d \in \mathcal{D},\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}. \tag{3.20i}$$

For the formulation of crop- and depot-dependent cost coefficients, the minimum and maximum active crop-indices need to be identified. Let therefore $\alpha^k \in \{0,1\}$ indicate if crop k is active in the sense of $\alpha^k = 1$ if $\sum_{l \in \mathcal{L}} \delta_l^k \geq 1$. By (1.9), this translates to (3.19f) and (3.19g). We introduced auxiliary variables $\tilde{\alpha}^k, \tilde{\beta}^k \in \{0,1\}$ indicating if crop k is the *smallest*- or *largest*-indexed active crop, respectively ($k = k^{\min}$ and $k = k^{\max}$). It holds that $\sum_{k \in \mathcal{K}} \tilde{\alpha}^k = 1$ and $\sum_{k \in \mathcal{K}} \tilde{\beta}^k = 1$. We then derive the nonlinear relations $\tilde{\alpha}^0 = \alpha^0$, $\tilde{\alpha}^1 = \alpha^1(1 - \tilde{\alpha}^0)$, $\tilde{\alpha}^2 = \alpha^2(1 - \tilde{\alpha}^1 - \tilde{\alpha}^0)$, …, which can be translated to

$$\tilde{\alpha}^0 = \alpha^0, \tag{3.21}$$

$$\alpha^k + (1 - \sum_{\tau=0}^{k-1} \tilde{\alpha}^\tau) - \tilde{\alpha}^k \leq 1,\ \forall k = 1,\dots,K-1, \tag{3.22}$$

$$\tilde{\alpha}^k \leq \alpha^k,\ \tilde{\alpha}^k \leq 1 - \sum_{\tau=0}^{k-1} \tilde{\alpha}^\tau,\ \forall k = 1,\dots,K-1. \tag{3.23}$$

Similarly, starting the iteration from highest $k = K-1$ with $\tilde{\beta}^{K-1} = \alpha^{K-1}$, we can derive nonlinear relations for $\tilde{\beta}^k$ to ultimately obtain (3.19j), (3.19k) and (3.19l). Suppose the *path-dependent* part of the cost function taking the *nonlinear* form $\sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{L}} \left(c_{dj}^{k,k^{\min}} \tilde{\alpha}^k + c_{dj}^k (1 - \tilde{\alpha}^k) \right) x_{dj}^k +$ $\sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{L}} \left(c_{jd}^{k,k^{\max}} \tilde{\beta}^k + c_{jd}^k (1 - \tilde{\beta}^k) \right) x_{jd}^k$ with $c_{dj}^{k,k^{\min}} \geq 0$ and

$c_{jd}^{k,k^{\max}} \geq 0$ denoting cost-coefficients that are distinct for the first (i.e., $k = k^{\min}$ or $\tilde{\alpha}^k = 1$) and last (i.e., $k = k^{\max}$ or $\tilde{\beta}^k = 1$) crop-route. Then, auxiliary variables $v_{dj}^k \in \{0,1\}$ and $w_{jd}^k \in \{0,1\}$ need to be introduced with $\sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}}\sum_{k\in\mathcal{K}} v_{dj}^k = 1$ and $\sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}}\sum_{k\in\mathcal{K}} w_{jd}^k = 1$. They are related according $v_{dj}^k = \tilde{\alpha}^k x_{dj}^k$ and $w_{jd}^k = \tilde{\beta}^k x_{jd}^k$, $\forall d \in \mathcal{D},\ j \in \mathcal{L},\ k \in \mathcal{K}$ and can be translated to integer linear inequalities according to (1.10). The objective function part above can now be expressed *linearly* dependent on decision variables, see (3.18a).

IP-5. We fix $m = K$. Thus, there is one decision variable less w.r.t. IP-1. IP-5 is identical to (3.14) with few exceptions: γm in (3.14a) is constant, (3.14i) can be omitted, and constraints (3.14d) are replaced by: $\sum_{j\in\mathcal{L}} x_{dj}^k = 2,\ \forall k \in \mathcal{K}$.

IP-6. We can adopt IP-5 with the exception of modified cost coefficients according to $c_{dj}^k = c_{dj}^{k,k^{\min}},\ \forall k \in \mathcal{K}$.

IP-7. We fix $m = K$ in IP-3. As a consequence, (3.17) is rendered linear and (3.16) simplifies to

$$\min \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{L}} c_{dj}^k x_{dj}^k + \sum_{k\in\mathcal{K}}\sum_{i<j} c_{ij}^k x_{ij}^k - \sum_{l\in\mathcal{L}}\sum_{k\in\mathcal{K}} r_l^k \delta_l^k + \gamma K + \sum_{d\in\mathcal{D}} n^d \eta^d \tag{3.24a}$$

$$\text{s.t.} \sum_{d\in\mathcal{D}} x_{dl}^k + \sum_{i<l} x_{il}^k + \sum_{l<j} x_{lj}^k = 2\delta_l^k, \begin{cases} \forall l \in \mathcal{L}, \\ \forall k \in \mathcal{K}, \end{cases} \tag{3.24b}$$

$$\sum_{k\in\mathcal{K}} \delta_l^k = 1,\ \forall l \in \mathcal{L}, \tag{3.24c}$$

$$\sum_{j\in\mathcal{L}} x_{dj}^k = 2\eta^d,\ \forall d \in \mathcal{D},\ \forall k \in \mathcal{K}, \tag{3.24d}$$

$$\sum_{d\in\mathcal{D}} \eta^d = 1, \tag{3.24e}$$

$$\sum_{i<j;i,j\in S^k} x_{ij}^k \leq |S^k| - 1,\ 3 \leq |S^k| \leq N-1, \quad \forall k \in \mathcal{K},\ S^k \subseteq \mathcal{V}\backslash\{d\},\ \forall d \in \mathcal{D}, \tag{3.24f}$$

with decision variables

$$x_{dj}^k \in \{0,1,2\},\ \forall d \in \mathcal{D},\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.25a}$$
$$x_{ij}^k \in \{0,1\},\ 0 \le i < j,\ \forall k \in \mathcal{K}, \tag{3.25b}$$
$$\delta_l^k \in \{0,1\},\ \forall l \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.25c}$$
$$\eta^d \in \{0,1\},\ \forall d \in \mathcal{D}. \tag{3.25d}$$

IP-8. Constraining $m = K$ significantly simplifies (3.18) to

$$\min \sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}} c_{dj}^{0,k_{\min}} x_{dj}^0 + c_{jd}^{K-1,k_{\max}} x_{jd}^{K-1} + \gamma K + \sum_{d\in\mathcal{D}} n^d \eta^d + \sum_{d\in\mathcal{D}}\sum_{k\in\tilde{\mathcal{K}}}\sum_{j\in\mathcal{L}} c_{dj}^k x_{dj}^k + \sum_{k\in\mathcal{K}}\sum_{i<j} c_{ij}^k x_{ij}^k - \sum_{l\in\mathcal{L}}\sum_{k\in\mathcal{K}} r_l^k \delta_l^k \tag{3.26a}$$

$$\text{s.t.} \sum_{d\in\mathcal{D}} x_{dl}^k + \sum_{i<l} x_{il}^k + \sum_{l<j} x_{lj}^k + \sum_{d\in\mathcal{D}} x_{ld}^k = 2\delta_l^k, \begin{cases} \forall l \in \mathcal{L}, \\ \forall k \in \mathcal{K}, \end{cases} \tag{3.26b}$$

$$\sum_{k\in\mathcal{K}} \delta_l^k = 1,\ \forall l \in \mathcal{L}, \tag{3.26c}$$

$$\sum_{j\in\mathcal{L}} x_{dj}^k + x_{jd}^k = 2\eta^d,\ \forall d \in \mathcal{D},\ \forall k \in \mathcal{K}, \tag{3.26d}$$

$$\sum_{j\in\mathcal{L}} x_{dj}^k = \eta^d,\ \sum_{j\in\mathcal{L}} x_{jd}^k = \eta^d,\ \forall d \in \mathcal{D},\ \forall k \in \mathcal{K}, \tag{3.26e}$$

$$\sum_{d\in\mathcal{D}} \eta^d = 1, \tag{3.26f}$$

$$\sum_{i<j;i,j\in S^k} x_{ij}^k \le |S^k| - 1,\ 3 \le |S^k| \le N-1, \quad \forall k \in \mathcal{K},\ S^k \subseteq \mathcal{V}\backslash\{d\},\ \forall d \in \mathcal{D}, \tag{3.26g}$$

$$x_{dj}^k \in \{0,1\},\ \forall d \in \mathcal{D},\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.26h}$$
$$x_{ij}^k \in \{0,1\},\ 0 \le i < j,\ \forall k \in \mathcal{K}, \tag{3.26i}$$
$$\delta_l^k \in \{0,1\},\ \forall l \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.26j}$$
$$\eta^d \in \{0,1\},\ \forall d \in \mathcal{D}, \tag{3.26k}$$
$$x_{jd}^k \in \{0,1\},\ \forall d \in \mathcal{D},\ \forall j \in \mathcal{L},\ \forall k \in \mathcal{K}, \tag{3.26l}$$

with $\tilde{\mathcal{K}} = \mathcal{K}\backslash\{0, K-1\}$.

Comparison of IP-formulations

Let us first compare above IP-formulations *coupling* crop assignment with routing vs. a *two-stage* approach with the first stage the solution of an assignment problem and the second stage the solution of one TSP for each crop. Any coupling approach is *always* at least as good as any two-stage method. This is since the optimal solution of the latter is always a feasible solution of the former method. Without making further assumption, no further general statements can be made. Under assumptions, simple heuristic algorithms based on inequality checks can be developed and applied to the two-stage solution to determine suboptimality w.r.t. the coupling solution.

Let us denote the objective value of IP-2 and IP-3 by $J_{\mathsf{IP}-2}$ and $J_{\mathsf{IP}-3}$, respectively.

Proposition 10. *It always holds that* $J_{IP-3} \leq J_{IP-2}$.

Proof. The proof is by contradiction. Let us assume $J_{\mathsf{IP}-2} < J_{\mathsf{IP}-3}$. $J_{\mathsf{IP}-2}$ and $J_{\mathsf{IP}-3}$ differ by cost coefficients $c_{dj}^{k} = c_{dj}^{k,k_{\min}},\ \forall k \in \mathcal{K}$ and c_{dj}^{k} in (3.8), respectively. By linearity of $J_{\mathsf{IP}-2}$ and the definition of $c_{dj}^{k,k_{\min}}$ according to (3.9), and by nonnegativity of $\tilde{c}_{\tilde{d}j}^{k}$, $J_{\mathsf{IP}-2}$ can always be lowered by concentrating all harvesters, $\sum_{\tilde{d}\in\mathcal{D}} N_{\tilde{d}}^{\mathrm{harv},k}$, to the most cost-efficient depot. This is the IP-3 solution and therefore contradicts our assumption. The equality-part is because a special case of IP-2 is that none harvesters are initially located at any of the depots except the optimal one according to IP-3. This concludes the proof. □

It always is $J_{\mathsf{IP}-3} \leq J_{\mathsf{IP}-1}$ since the latter single depot case is always included in the former multiple depot case.

Generalizing statements regarding $J_{\mathsf{IP}-1}$ vs. $J_{\mathsf{IP}-2}$, and likewise for $J_{\mathsf{IP}-3}$ vs. $J_{\mathsf{IP}-4}$ cannot be made. This is because it is always possible to create counterexamples in favor of one or another solution.

It always is $J_{\mathsf{IP}-n} \leq J_{\mathsf{IP}-(n+4)},\ \forall n = 1, \ldots, 4$. This is because of the greater freedom in *not* having to use *all* crops for the final solution for the first four cases.

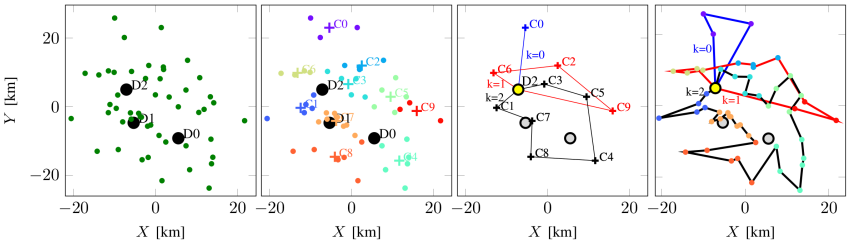


Figure 65: Illustration of Steps 1 to 4 (Plot 1 to 4 from left to right) of Algorithm 14. (Plot 1) Three depots (D0, D1 and D2) and 50 fields are visualized by the black and green balls, respectively. (Plot 2) The fields are assigned to $\tilde{k} = 10$ clusters (C0,...,C9). All fields belonging to the same cluster are colored correspondingly. The cross-signs indicate the $\tilde{k}$ centroids and are labeled accordingly. (Plot 3) Results of IP-7 applied to the $\tilde{k}$ centroids. (Plot 4) Results of CApR-7. For visualization, the fields are colored according to the clustering result. Labels $k = 0$, $k = 1$ and $k = 2$ indicate the *first* traversal of each crop-tour, whereby a *crop-tour* denotes the harvesting routes associated with a specific crop k.

Main Algorithm

The main algorithm is summarized in Algorithm 14. See Figure 65 for its visualization. It is used for *crop assignment plus routing* (CApR). We denote the reversing of a list or sequence of elements by the flip$(\cdot)$-operator. Several remarks can be made.

First, Algorithm 14 is motivated to handle a large number of fields. This is achieved by the proposed layered approach. It comes, however, at the cost of returning, in general, a suboptimal solution. The exact and global optimum solution is attained for $\tilde{k} = L$. In practice, it is recommended to increase the number of clusters as much as computational power and available IP-solver permit, ideally, until $\tilde{k} = L$ such that Steps 4-13 can be omitted entirely.

Second, the relations between various $J_{\mathsf{IP}-n}$ for all $n = 1, \ldots, 8$, can in general *not* be translated to the corresponding objective values of the CApR-n solutions. This is because of the *heuristic* (layered) nature of Algorithm 14. For instance, in general, it does not always hold that

Algorithm 14: CApR-n

Input: $\{P_l\}_{l=0}^{L-1}$, $\{P_d\}_{d=0}^{D-1}$, c_{dj}^k, c_{ij}^k, $c_{dj}^{k,k_{\min}}$, $c_{jd}^{k,k_{\max}}$, c_{jd}^k, r_l^k, γ, $\{\eta^d\}_{d=0}^{D-1}$ and $\tilde{k}$.

Clustering:

- cluster L fields spatially according to $\tilde{k}$-means [173].
- let the sets of fields associated with each cluster be denoted by $\mathcal{L}_{z,\xi} \subset \mathcal{L},\ \forall \xi = 0, \dots, \tilde{k}-1$.
- let the set of clusters be denoted by $\mathcal{L}_z$ with $|\mathcal{L}_z| = \tilde{k}$.
- assign a coordinate $P_{l_z} \in \mathbb{R}^2,\ \forall l_z \in \mathcal{L}_z$, to each cluster (the centroids).
- compute $c_{dj_z}^k$, $c_{i_z j_z}^k$, $c_{dj_z}^{k,k_{\min}}$, $c_{j_z d}^{k,k_{\max}}$, $c_{j_z d}^k,\ \forall i_z, j_z \in \mathcal{L}_z$.
- compute $r_{l_z}^k = \sum_{l \in \mathcal{L}_{z,\xi}} r_l^k,\ \forall l_z \in \mathcal{L}_z,\ \xi = 0, \dots, \tilde{k}-1$.

Integer Programming (IP-n):

- solve IP-n from Section 3.5.4 for the clustering result of Step 2, replacing $\mathcal{L}$ by $\mathcal{L}_z$ and cost coefficients accordingly.
- let the resulting set of active crops and optimal basis depot be denoted by $\mathcal{M}^\star \subseteq \mathcal{K}$ and $d^\star \in \mathcal{D}$, respectively.
- let $\mathcal{C}^k$ denote the sequence of clusters $\forall k \in \mathcal{M}^\star$, whereby every sequence starts and ends at $d^\star \in \mathcal{D}$.

From Cluster- to Field-sequences: **for** $k \in \mathcal{M}^\star$ **do**

- define $\mathcal{C}^{k,1} = \mathcal{C}^k$ and $\mathcal{C}^{k,2} = \text{flip}\left(\mathcal{C}^k\right)$.
- **for** $i = 1, 2$ **do**
  - **for** $\mathcal{C}^{k,i}$ **do**
    - find closest fields between any pair of consecutive clusters $c^{(t)}, c^{(t+1)} \in \mathcal{C}^{k,i}$ within the $\mathcal{C}^{k,i}$-tour, where $t = 0, \dots, |\mathcal{C}^{k,i}|$.
    - let the two fields associated with each cluster $c^{(t)}$ be denoted by $s^{(t)}$ and $e^{(t)}$.
    - for each cluster $c^{(t)},\ \forall t$, solve a TSP connecting $s^{(t)}$ and $e^{(t)}$ to obtain a corresponding field-sequence $f^{(t)} = \{s^{(t)}, \dots, e^{(t)}\}$.
    - concatenate all field-sequences to crop-tour $\mathcal{F}^{k,i} = \{f^{(0)}, \dots, f^{(|\mathcal{C}^{k,i}|)}\}$ and determine its path length $d^{k,i}$.
  - **if** $d^{k,1} < d^{k,2}$ **then**
    - $\mathcal{F}^{k,\star} = \mathcal{F}^{k,1}$, else $\mathcal{F}^{k,\star} = \mathcal{F}^{k,2}$.

Output:

- set of active crops $\mathcal{M}^\star$ and basis depot selection $d^\star \in \mathcal{D}$.
- crop assignment to every field, $\delta_l^{k,\star},\ \forall l \in \mathcal{L}, \forall k \in \mathcal{M}^\star$.
- crop-tour $\mathcal{F}^{k,\star},\ \forall k \in \mathcal{M}^\star$.

Algorithm 15: Renting out and Taking Leases

1 Define all fields considered by $\mathcal{L} = \mathcal{L}^{\text{own}} \cup \mathcal{L}^{\text{ptl}}$.
2 Define the set of fields of interest by $\tilde{\mathcal{L}} = \mathcal{L}^{\text{pro}} \cup \mathcal{L}^{\text{ptl}}$.
3 Modeling according farmer's *own* production means.
- determine parameters of Step 1 of Algorithm 14, $\forall l \in \mathcal{L}$.

4 **Solve** a *relaxed* CApR-n for any desired $n = 1, \ldots, 8$.
5 Determine $\mathcal{L}^{\text{ntl}} = \{l \in \mathcal{L}^{\text{ptl}} : \delta_l^k = 0, \forall k \in \mathcal{K}\}$.
- *not* take a lease on any of these fields.

6 Determine $\mathcal{L}^{\text{ro}} = \{l \in \mathcal{L}^{\text{pro}} : \delta_l^k = 0, \forall k \in \mathcal{K}\}$.
- rent out all of these fields (*any* positive return is good).

7 **Solve** standard CApR-n for $\mathcal{L}^1 = \mathcal{L} \backslash \{\mathcal{L}^{\text{ntl}} \cup \mathcal{L}^{\text{ro}}\}$.
- denote its objective value by $J_{\mathcal{L}^1}$.

8 **Solve** standard CApR-n for $\mathcal{L}^{\text{own}}$.
- denote its objective value by $J_{\mathcal{L}^{\text{own}}}$.

9 Take leases of fields $\mathcal{L}^{\text{ptl}} \backslash \mathcal{L}^{\text{ntl}}$ for the *overall* payment rate of at most $\Delta J = J_{\mathcal{L}^1} - J_{\mathcal{L}^{\text{own}}}$.

$J_{\text{CApR}-3} \leq J_{\text{CApR}-2}$.

Third, under the absence of priority constraints according to Section 3.5.3, there exist two directions in which to traverse any crop-tour. The traversal direction affects the closest fields between any pair of consecutive clusters. Consequently, the TSP-solution for each cluster, and thereby ultimately the total path length of the crop-tour, is affected, too. This motivated to test *both* cluster-sequences as indicated in Step 4. As stated, Algorithm 14 does not account for priority constraints, i.e., for a priori modeling of field ripeness sequences. Therefore, Step 2 and 4 require modification and clustering must be conducted according to an objective accounting for ripeness level. As a consequence, the traversal direction for Step 4 would also be fixed.

Fourth, the result of Algorithm 14 could in principle be further refined by heuristic local searches such as, for example, the local exchange of field-pairs within a crop-tour sequence if it improves the CApR-n objective function value. Naturally, at this stage, local field sequences can

also be exchanged manually by farm operators.

3.5.5 Extensions

Financial Considerations Regarding Leasing

The clustering step of Algorithm 14 does not necessarily have to be conducted according to spatial proximity of fields. Fields can be clustered arbitrarily. Also, single fields can be assigned to a single cluster for special analysis. For leasing considerations, the *partial* service of a subset of fields is of interest. Let subset $\tilde{\mathcal{L}} \subseteq \mathcal{L}$ denote all fields for which we do not necessarily want to enforce field service but contemplate leasing options. Then, for IP-3, we maintain equality constraints (3.16b) and (3.16c) only for $\mathcal{L} \backslash \tilde{\mathcal{L}}$, and define relaxed inequalities

$$\sum_{d \in \mathcal{D}} x_{dl}^k + \sum_{i<l} x_{il}^k + \sum_{l<j} x_{lj}^k \leq 2\delta_l^k, \forall l \in \tilde{\mathcal{L}}, \forall k \in \mathcal{K}, \tag{3.27}$$

$$\sum_{k \in \mathcal{K}} \delta_l^k \leq 1, \ \forall l \in \tilde{\mathcal{L}}. \tag{3.28}$$

We similarly relax corresponding constraints for all other IP-n. Any CApR-n including such constraints, shall be denoted as *relaxed* CApR-n. In constrast, the original problem according to Algorithm 14 is referred to as standard CApR-n.

An important financial consideration for every farm is to decide upon *either* servicing *or* renting out of one's fields, and additionally the decision upon taking of leases on additional fields for coverage. Let us denote the sets of corresponding fields by $\mathcal{L}^{\text{own}}$ (farmer's own fields), $\mathcal{L}^{\text{pro}} \subset \mathcal{L}^{\text{own}}$ (potential rent outs) and $\mathcal{L}^{ptl}$ (potential fields for taking leases upon), respectively. Then, Algorithm 15 provides guidelines for decision making. Let us elaborate on Steps 4-6 of Algorithm 15. Suppose a field does not improve the total financial return, typically, because of too expensive production costs (consider, for example, fields very distant apart from depots) or constraints such as (3.15) for limited harvesting windows. Then, renting out is profitable, in theory, already for any positive return. In practice, the farmer is naturally advised to negotiate renting out rates as favorably as possible.

Let us also discuss Step 9 of Algorithm 15. In contrast to pure assignment problems, the maximum leasing rate ΔJ cannot easily be distributed among corresponding fields. This is because monetary profits are nonlinearly related to crop returns because of the coupling with routing decisions. Importantly, the precise distribution of leasing rates of individual fields is not relevant as long as it *overall* does not surpass ΔJ. Thus, ΔJ provides the farmer with an *upper bound* on profitable leasing rates. If ΔJ cannot be attained in negotiations, different $\mathcal{L}^{\text{ptl}}$ should be decided and Algorithm 15 solved again. This is repeated until a corresponding upper bound can be satisfied, or, ultimately, $\mathcal{L}^{\text{own}} \backslash \mathcal{L}^{\text{ro}}$ are serviced.

The second financial consideration is motivated by the comparison of objective values for CApR-n. It permits to determine "fair" prices for leasing when sheltering machinery at the various depots. It is envisioned that *all* collaborating farmers first involve in accurate system modeling (cost coefficients), before then solving either all of CApR-n, $\forall n = 1, \ldots, 4$, or all of CApR-n, $\forall n = 5, \ldots, 8$. Specifically, the difference in objective values between CApR-2 (or CApR-7 for enforcement of all K crops in the solution) and the remaining CApR-n then permits to determine an *upper bound* on leasing rates for depot usage.

Application in Practice

For operations planning in practice, detailed modeling of the parameters listed in Step 1 of Algorithm 14 is of paramount importance for optimal results. Historical field and crop yield data must serve as basis. By the selection of c_{ij}^k, computational complexity can be reduced by pruning specific undesired field connections from a path network, thereby implicitly also influencing priority constraints. Large fields often have multiple possible field entrance and exit points. This may significantly affect travel distances between fields. In fact, field coverage patterns and in-field navigation [199], [84], [149] can also be co-planned a priori to account for crop-tours efficiently linking fields planting the same crops. This is subject of ongoing work.

Table 8: Experiment 1. The percentage out of the 50 simulation experiments for which an IP-n solution could be found in less than 200 SEC-iterations is denoted by $P_{\text{conv}}^{\mathsf{IP}-n}$. The absolute average monetary objective value is denoted by $\bar{J}^{\mathsf{CApR}-n}$. The average number of decision variables, SEC-iterations, number of equality constraints, inequality constraints when first omitting SECs and for the final SEC-iteration (before convergence) are denoted by $N_z^{\mathsf{IP}-n}$, $\bar{N}_{\text{iterSEC}}^{\mathsf{IP}-n}$, $N_{\text{eq,NoSEC}}^{\mathsf{IP}-n}$, $N_{\text{ineq,NoSEC}}^{\mathsf{IP}-n}$ and $\bar{N}_{\text{ineq,finalIP}}^{\mathsf{IP}-n}$, respectively. Average combined CPU-time for the solution of all SEC-iterations is $\bar{T}_{\text{CPU}}^{\mathsf{IP}-n}$.

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$	unit
$\bar{J}^{\mathsf{CApR}-n}$	746766.5	743866.0	748922.1	745887.6	744816.7	743201.4	746945.8	745970.1	€
$\bar{T}_{\text{CPU}}^{\mathsf{IP}-n}$	7.41	17.32	16.98	3.00	0.07	0.05	0.10	0.14	s
$N_z^{\mathsf{IP}-n}$	196	196	262	541	195	195	258	348	–
$\bar{N}_{\text{iterSEC}}^{\mathsf{IP}-n}$	24.78	37.96	22.88	1.78	2.62	2.18	2.32	1.80	–
$N_{\text{eq,NoSEC}}^{\mathsf{IP}-n}$	41	41	44	68	43	43	50	68	–
$N_{\text{ineq,NoSEC}}^{\mathsf{IP}-n}$	197	197	275	1112	195	195	258	348	–
$\bar{N}_{\text{ineq,finalIP}}^{\mathsf{IP}-n}$	226.04	240.49	301.50	1112.88	197.00	196.36	259.56	348.90	–
$P_{\text{conv}}^{\mathsf{IP}-n}$	98	90	100	100	100	100	100	100	%

Table 9: Normalized average monetary returns in Northern Germany.

	barley $k=0$	rapeseed $k=1$	wheat $k=2$	unit
$\tilde{r}^k$	570	600	750	€/ha

3.5.6 Numerical Simulations

For the solution of integer programs, we employ the domain-specific language CVXPY for optimization embedded in Python [92] with default settings. All experiments were conducted on a laptop running Ubuntu 16.04 with Intel Core i7 CPU @2.80GHz×8 and 15.6GB of memory.

Problem data is generated randomly with realistic parameter settings from farming in Northern Germany. For the first numerical simulation experiments, we assume three depots, a maximal number of three crops and 50 fields, i.e., $D = 3$, $K = 3$ and $L = 50$. Fields are clustered spatially into sets according to Step 2 of Algorithm 14, whereby we selected $\tilde{k} = 10$. Field and depot locations are generated randomly according to a Gaussian distribution centered at the origin with standard deviations $\sigma_d = 10$km, $\forall d \in \mathcal{D}$, and $\sigma_l = 15$km, $\forall l \in \mathcal{L}$. To each depot, we randomly

assign a number of harvesters according $N_d^{\text{harv},k} = \max(1, \lfloor 5u_d \rfloor),\ u_d \sim \mathcal{U}(0,1),\ \forall d \in \mathcal{D}$, where $\mathcal{U}(0,1)$ denotes the Uniform distribution with zero mean and unit variance, and $\lfloor \cdot \rfloor$ denotes rounding to the next smallest integer. Normalized traveling costs per harvester and km are set as $\tilde{c} = 30\frac{€}{\text{km}}$. We assume a cost of $\gamma = 1000€$ for every planted crop. Here, maintenance costs are assumed to be identical for all depots. W.l.o.g., we therefore set $n^d = 0,\ \forall d \in \mathcal{D}$. Realistic normalized monetary returns in € per ha and crop are determined as mean values from intermediate soil qualities and crop yields in Northern Germany. They are summarized in Table 9. Regarding the monetary return per field and crop, we considered two settings. First, we generate field sizes in hectares according to $s_l = \max(20 + 10u_d, 1),\ u_d \sim \mathcal{N}(0,1),\ \forall l \in \mathcal{L}$, where $\mathcal{N}(0,1)$ denotes the Gaussian distribution with zero mean and unit variance. In combination with $L = 50$ this results approximately in a total coverage size of 1000ha. According to survey [346], in *all* of Germany there are 299134 farming businesses of which only 1502 have a size of more than 1000ha. The field sizes are then multiplied with $\tilde{r}^k$ according to Table 9, to yield $r_l^k = s_l \tilde{r}^k,\ \forall l \in \mathcal{L},\ k \in \mathcal{K}$. This method of data generation is intuitive. Since normalized monetary return is considerably higher for wheat than for barley and rapeseed, the application of Algorithm 14 typically assigns wheat to *all* fields, unless crop rotation constraints, or diversification constraints, as in CApR-n for $n = 5, \ldots, 8$, are included. In the latter cases, the crop with smallest monetary return is assigned to the cluster with smallest field area, and the crop with second-smallest return to the second-smallest area and so forth. In a second setting, and to add more variety, we therefore generated monetary returns per field and crop according to

$$r_l^k = \max(20 + 10u_l^k, 1)\tilde{r}^k,\ \forall l \in \mathcal{L},\ k \in \mathcal{K}, \tag{3.29}$$

with $u_l^k \sim \mathcal{N}(0,1)$. To analyze computational aspects of the proposed algorithm, we analyzed 50 random data sets, generated as outlined above. The results are summarized in Table 8. Several observations can be made. First, *fixing* the number of serviced crops, as for CApT-n for $n = 5, \ldots, 8$, significantly reduces CPU-time $\bar{T}_{\text{CPU}}$, by (on average) more than two or-

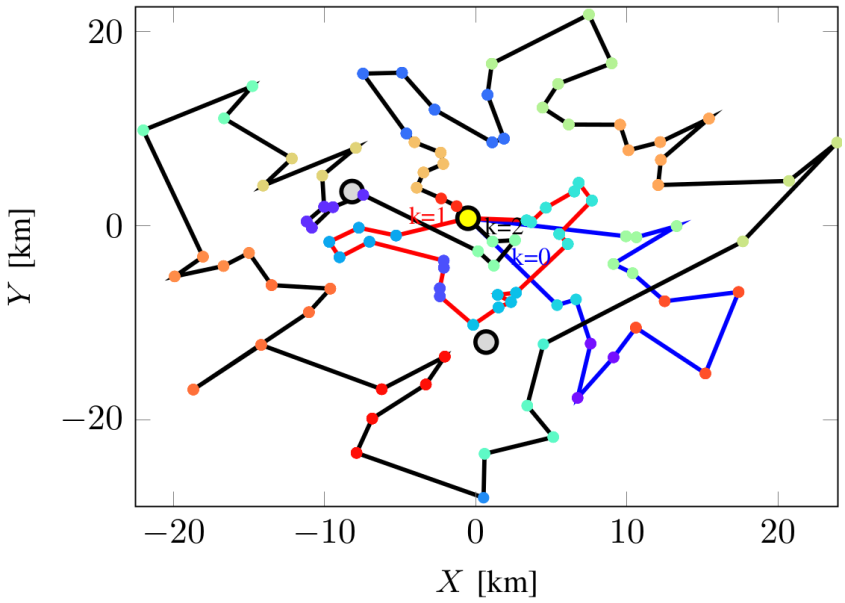


Figure 66: Visualization of an example for $\tilde{k} = 25$ and $L = 100$ fields. The results of CApR-7 are displayed. The inactive depots and the active depot are denoted by the larger gray and yellow balls, respectively. The fields are colored according to their clustering result. Labels $k = 0$, $k = 1$ and $k = 2$ indicate the first edge traversal of each crop-tour.

ders of magnitude. The main computational burden stems from an increased number of SEC-iterations. For every additional SEC-iteration, an additional IP with an increased number of SECs has to be solved. IP-2 had difficulties converging within 200 SEC-iterations in some cases.

We tested problem instances with up to 35 clusters and 700 fields. For $\tilde{k} > 35$, CVXPY started to fail to converge. Real-world practical cases in Northern Germancy for the collaboration of larger farms may involve around 100 fields and three depots. A corresponding simulation experiment is visualized in Figure 66. For its solution with 1119 integer variables, 106 equality constraints and between 1119 (for the first IP-solution without any SECs) to 1130 (for the last SEC-iteration) inequality constraints, 8 SEC-iterations were required with a total CPU-time of 2.6s.

3.5.7 Conclusion

We presented a flexible framework for the coupling of crop assignment with vehicle routing for harvest planning in agriculture. The discussed problem is relevant since the decision upon crop assignment must be addressed by every farm manager at the beginning of every work-cycle. We compared eight different IP formulations. We found the four cases with enforced inclusion of any crop out of a set of crops to be computationally most efficient. This enforcement is applicable in practice since the list of eligible crops typically is very limited. For large-scale applications where sole IP formulations are not tractable anymore, we proposed a heuristic algorithm combining the IP-formulations with clustering of fields and the solution of local TSPs. To summarize, for practical applications, we thus recommend:

- For reasons of computational efficiency, focus on CApR-n for $n = 5, \ldots, 8$.
- Increase the number of clusters $\tilde{k}$ as much as the available combination of computational hardware and IP-solver permits in order to reduce the heuristic nature of Algorithm 14.
- Solve Algorithm 15 to determine rates for the renting out and taking of leases on fields.
- In case of a single depot, solve CApR-5.
- In case of multiple depots, always solve at least CApR-6, and additionally at least one of CApR-7 and CApR-8 to determine leasing rates for depot usage.
- Put emphasis on detailed parameter modeling according to Step 1 of Algorithm 14, and the inclusion of constraints on *crop rotation* (3.11) and *traveling time* (3.15).

Future work will include the testing of alternative IP-solvers including Numberjack [177] and [1], and reformulations of the SECs, for example, in form of MTZ-SECs [277] which introduce additional *continuous*

variables for SECs and thereby render the problem of *mixed* integer nature. It is also planned to test a *Tabu search* heuristic [127]. We stress that limited harvesting time windows with limited harvester traveling speeds naturally constrain the maximum number of fields that can be serviced within a crop-tour. Thus, more efficient IP-solvers are not sought to increase the maximum field number beyond 700, but to increase tractable $\tilde{k}$ and thereby reduce the heuristic nature of Algorithm 14. While this work focused on the development of a framework for planning at the *beginning* of the yearly work cycle, future work will also revise online coordination of machinery at the *end* of the work-cycle during harvest.

3.5.8 Hierarchical Controller Parametrization

There are three hierarchical layers in Algorithm 14. These are:

1. Clustering of fields according to spatial proximity;
2. The solution of one out of eight proposed IPs;
3. The relation of field clusters to IP-solutions by means of local TSPs.

3.6 Discussion of Chapter

Summary

The two main contributions [146] and [144] are complementary. The former is concerned about edge coverage and *in-field* logistics. In contrast, the latter is concerned about assignments and linking of nodes in the vehicle routing framework, which can be classified as *out-field* logistics. The former work is built from scratch and scales quite well because of the particular graph structure for connected field lane-segments, which can be exploited for the problem solution. In contrast, the latter work incorporates an IP-solver for its solution. It was found that this greatly limits scalability and was the reason for the introduction of the hierarchical implementation scheme. In particular, it necessitated the inclusion of a clustering step.

Future Work

The last remark made is a starting point for possible future work. Namely, the IP-solver must be replaced. Here, an algorithm optimizing via simulation may be a better choice. In fact, when using proposed integer programs as parametrizations and adapting the parameter perturbation Step 8, the TSHC-Algorithm 4 could be used for optimization via simulation. This may be subject of future work.

Chapter 4

Quantitative Finance

Quantitative finance represents an ideal testbed for stochastic optimization. This is for two reasons. First, a high degree of stochasticity and unpredictability is involved, which is the result of the plethora of different influencing factors. These include, on the one hand, "hard" influences such as earning reports, and, on the other hand, "soft" influences caused by emotions and the psychology of market participants. The latter phenomenon is broadly summarized under the term *behavioral finance* [108]. Second, there are many different sources of data available, ranging from historical price data and trading volume to news and social media. Here, currencies, stocks and options were considered as financial contracts in which investment positions can be taken.

4.1 Dynamic Option Hedging with Transaction Costs: A SMPC approach

This section summarizes [157]:

- M. Graf Plessen, L. Puglia, T. Gabbriellini, and A. Bemporad, "Dynamic option hedging with transaction costs: A stochastic model predictive control approach," *International Journal of Robust and Nonlinear Control*, pp. 1-20, 2017.

Stochastic model predictive control (SMPC) is proposed as a tool for hedging derivative contracts (such as plain vanilla and exotic options) in the presence of transaction costs. The methodology combines stochastic scenario generation for the prediction of asset prices at the next rebalancing interval with the minimization of a stochastic measure of the predicted hedging error. We consider three different measures to minimize in order to optimally rebalance the replicating portfolio: a trade-off between variance and expected value of hedging error, conditional value at risk (CVaR), and the largest predicted hedging error. The resulting optimization problems require solving at each trading instant a quadratic program (QP), a linear program (LP), and a (smaller scale) LP, respectively. These can be combined with three different scenario-generation schemes: the log-normal stock model with parameters recursively identified from data, an identification method based on support vector regression (SVR), and a simpler scheme based on perturbation noise. The hedging performance obtained by the proposed SMPC strategies is illustrated on real-world data drawn from the NASDAQ-100 composite, evaluated for a European call and a barrier option, and compared to delta-hedging.

4.1.1 Introduction

For a financial institution, hedging a derivative contract implies to dynamically rebalance a (self-financing) portfolio of underlying assets at periodic intervals so that, at the expiration date of the contract, the value of the portfolio is as close as possible to the payoff value to pay to the

customer. In contrast, static hedging strategies do not involve rebalancing; thus, a replicating portfolio is formed initially and simply let evolve freely for the whole option life. For general background on options and derivative contracts see for instance [191].

The most common derivative contracts are *plain vanilla* options: a European *call (put) option* gives the holder the right to buy (sell) the underlying asset at a given *expiration date* and at a determined *strike price*. A large number of other more complex derivative contracts, called *exotic* options, are nowadays traded, especially in the *over-the-counter* (OTC) market. An example of an exotic option is the *barrier option*, a special kind of plain vanilla contract whose payoff drops to zero as soon as the price of the underlying asset reaches a certain barrier value. See [150] for optimization-based combinations of options.

The most common approach used in practice to dynamically rebalancing the portfolio replicating the option is *delta-hedging* (Δ-hedging), which directly derives from the fundamental theory of [49]. In delta-hedging, the replicating portfolio includes a cash position and a quantity of stocks equal to the derivative of the option price with respect to the price of the underlying stock. The guiding notion of Δ-hedging is to make the portfolio insensitive to the stochastic evolution of the price of the underlying asset. In control theoretical words, this is equivalent to making the wealth of the portfolio tracking the option price while rejecting the disturbance induced by price fluctuations. Within Black-Scholes theory, Δ-hedging makes the following (often unrealistic) assumptions: the underlying price follows the *log-normal* stock model, continuous-time hedging, static volatility, and the *absence of transaction costs*.

To handle transaction costs [110] and [138] proposed analytic methods based on stochastic optimization. In [110] the option price and the optimal trading strategy are jointly determined to reduce the total risk of writing the option. In [138] a trinomial process is used for generating the scenarios required to setup a stochastic control problem, in which the objective function is the expected value of a given performance index. To cope with transaction costs, in [310] the hedging problem is formulated as a linear quadratic regulation (LQR) problem penalizing transaction

costs in the objective function. As an alternative, a MPC approach is proposed to solve a quadratic program over a specified horizon, exploiting the LQR solution from the first approach in the cost function. In [314] transaction costs are taken into account in a finite-horizon constrained stochastic control problem formulation that is iteratively solved at each trading date by employing a semi-definite programming algorithm. Related ideas proposing the use of MPC for replicating portfolios appeared earlier in [96], [182].

For the case *without* transaction costs, stochastic model predictive control (SMPC) approaches were proposed in [34], [35]. SMPC can be seen as a suboptimal way of solving a stochastic multi-stage dynamic programming problem: Rather than solving the problem for the entire remaining time-span of the option life, a smaller problem is solved repeatedly from the current time-step t up to a certain number N of time steps in the future, by suitably re-mapping the condition at the future expiration date into a value at the predicted time step $t + N$. Formulating the stochastic optimization problem requires enumerating a certain number of scenarios of future stock prices. A suitable stock-price model is not known a priori and its parameters must be identified from data.

Here, we propose SMPC to solve dynamical option hedging problems with transaction costs. We consider different performance measures (a trade-off between variance and expected value of hedging error, conditional value at risk (CVaR), and the largest predicted hedging error) and show how the corresponding optimization problems can be easily solved via either quadratic or linear programming. We largely extend [42] by considering real-world data in our results, drawn from the NASDAQ-100 composite, and by discussing various scenario-generation schemes for both step-ahead stock and option prices to construct the stochastic optimization problems.

Comments on the Risk-free Rate

We denote the *effective annual risk-free rate* by r_a. Assuming $T_{\text{td}} = 252$ *trading days* and a sampling step of $T_s = 1\text{day}$, we here compute the (daily) *risk-free rate* r by solving $(1 + r)^{T_{\text{td}}} = 1 + r_a$. In contrast, the

continuously compounded risk-free rate r_c is computed from $e^{r_c T_{\text{td}}} = 1 + r_a$. The option life T may be $T = T_{\text{td}}$, but does not necessarily have to.

4.1.2 Dynamic Option Hedging

Consider the problem of hedging an option $\mathcal{O}$ defined over n underlying assets. We denote by T_s the time interval between two consecutive trading dates (the results below can be easily generalized to non-uniform trading intervals T_s), by t the trading instants, $t = 0, 1, \ldots, T$, and by $s(t) = [s_1(t) \ \ldots \ s_n(t)]^{\mathsf{T}} \in \mathbb{R}^n$ the vector of spot prices of the assets.

In general, the option price $p(t)$ of $\mathcal{O}$ at a generic instant t is the discounted expectation of the payoff $\mathcal{P}(m(T))$ at expiration date in the risk-neutral measure, given the market state $m(t)$ at time t ($m(t) = s(t)$ for plain vanilla options). Denoting by T the maturity of $\mathcal{O}$ in terms of number of sampling steps of duration T_s, the price of the hedged option at a generic intermediate date tT_s is $p(t) = (1+r)^{t-T}\mathrm{E}\left[\mathcal{P}(m(T))|m(t)\right]$, where $\mathrm{E}[p(T)]$ is the expected value of the payoff in the risk-neutral measure. For European call options the payoff is

$$\mathcal{P}(m(T)) = p(T) = \max(s(T) - K, 0), \tag{4.1}$$

while for barrier options it is

$$\begin{aligned} p(T) &= \begin{cases} \max(s(T) - K, 0) & \text{if } s(t) < s_u, \ \forall t \leq T \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} \max(s(T) - K, 0) & \text{if } s_\ell(t) = 0 \\ 0 & \text{if } s_\ell(t) = 1, \end{cases} \end{aligned} \tag{4.2}$$

where s_u define the upper barrier level, and $s_\ell(t) \in \{0, 1\}$ is a logic state with dynamics $s_\ell(t+1) = s_\ell(t)$ OR $[s(t) \geq s_u]$, $s_\ell(0) = 0$ (in this case $m(t) = \{s(t), s_\ell(t)\}$).

Assume that there are no transaction costs, and that the standard *self-financing* condition holds, i.e., that the wealth $w(t)$ of the portfolio replicating option $\mathcal{O}$ is always totally reinvested. Then, the dynamics of the wealth $w(t)$ of the portfolio is

$$w(t+1) = (1+r)w(t) + \sum_{i=1}^{n} b_i(t)u_i(t), \tag{4.3}$$

where $u_i(t)$ is the quantity of asset i held at time t and $b_i(t) \triangleq s_i(t+1) - (1+r)s_i(t)$ is the excess return, i.e., how much the asset gains (or loses) with respect to the risk-free rate. The initial condition $w(0)$ is set equal to the price paid by the customer to purchase option $\mathcal{O}$, $w(0) = (1+r)^{-T}\mathrm{E}[p(T)|m(0)]$.

Dynamic hedging aims at making the final wealth $w(T)$ as close as possible to $p(T)$ for all possible market realizations. The hedging problem can be restated as a stochastic control problem. Using control systems jargon, the wealth $w(t) \in \mathbb{R}$ represents the "state" and the "regulated output" of the controlled process, traded asset quantities $u(t) \in \mathbb{R}^n$ are the "inputs", the option price $p(t)$ is the "reference" for $w(t)$. By defining the "tracking error" $e(t) \triangleq w(t) - p(t)$, the objective can be restated as the one of minimizing $e(t)$ for all possible asset price realizations.

As shown in [34], [35], in the absence of transaction costs and under the lack of arbitrage, a way to achieve this is to minimize the variance of the hedging error

$$J(e(T)) = \mathrm{E}\left[(e(T) - \mathrm{E}[e(T)])^2\right] \tag{4.4}$$

by solving the one-step ahead minimum-variance problem

$$\min_{\{u(t)\}} \quad \mathrm{Var}_{m(t+1)}\left[w(t+1, m(t+1)) - p(t+1, m(t+1))\right] \tag{4.5a}$$

$$\text{s.t.} \quad w(t+1, m(t+1)) = (1+r)w(t) + \sum_{i=0}^{n} b_i(t, m(t+1))u_i(t) \tag{4.5b}$$

with respect to the portfolio composition $u(t)$ at each trading date tT_s.

Note that expectations and variances are conditioned to the particular market realization $m(t)$ at time t; we omit here the conditional notation for simplicity and use the notation $w(t+1)$ since now on as a shortcut for the future wealth $w(t+1, m(t+1))$.

The formulation in (4.5) is equivalent to a stochastic model predictive control formulation with prediction horizon $N = 1$, under the terminal condition of perfect hedging between the prediction step $t+N$ and expiration step T. Problem (4.5) can be solved by enumerating a number

M of scenarios, each one corresponding to a different realization of a certain sequence of prices, and optimize the resulting sample variance. Each scenario j has probability π_j of occurring, $j = 1, \ldots, M$, $\pi_j > 0$, $\pi_j \leq 1$, $\sum_{k=1}^{M} \pi_j = 1$. Scenarios can be generated via Monte Carlo simulation [34], where $\pi^j = \frac{1}{M}$, or by discretizing a given probability density function that describes the disturbance process generating the asset prices [35]. Note that, contrarily to multi-stage stochastic programming approaches that typically limit the number M of considered scenarios to only 2 or 3 to avoid the combinatorial explosion over the optimization horizon N, here M can be quite large without incurring into prohibitive computation efforts, as the prediction horizon is simply $N = 1$.

By optimizing the sample variance of $w(t+1)-p(t+1)$, in the absence of transaction costs problem (4.5) can be rewritten as the following *least squares* problem

$$\min_{u(t)} \sum_{j=1}^{M} \pi^j \left(w^j(t+1) - p^j(t+1) - \left(\frac{1}{M} \sum_{i=1}^{M} w^i(t+1) - p^i(t+1) \right) \right)^2, \tag{4.6}$$

where $w^j(t+1) = (1+r)w(t) + \sum_{i=0}^{n} b_i^j(t) u_i(t)$ are the future values of portfolio wealth for each scenario $j = 1, \ldots, M$, and π^j is the corresponding probability, $\pi^j \geq 0$, $\sum_{i=1}^{M} \pi^j = 1$. The resulting SMPC algorithm is described by Algorithm 16.

Algorithm 16: SMPC algorithm for dynamic option hedging

1 Let t=current hedging date, $w(t)$= current wealth of portfolio, $m(t)$=current market state;
2 Generate M scenarios of future market states $m^1(t+1)$, …, $m^M(t+1)$, with corresponding probabilities π^1, …, π^M;
3 Use a pricing engine to generate the corresponding future option prices $p^1(t+1)$, …, $p^M(t+1)$;
4 Solve the least square problem (4.6) to minimize the sample variance of $w(t+1) - p(t+1)$;
5 Rebalance the portfolio according of the optimal solution $u^*(t)$ of problem (4.6);

An option-pricing engine is needed to compute the future option prices $p^1(t+1), \ldots, p^M(t+1)$. This is the most time-consuming operation of the entire algorithm. In fact, numerical pricing engines must be used, based on either Monte Carlo simulation, or on other approximate methods such as the method described by [259]. See [34], [35] for a comparison of different pricing methods. In particular, [35] showed that SMPC is superior to Δ-hedging when dealing with exotic options and quite robust also to errors in the dynamical model of the market, whereby all stock price data was generated artificially according to Heston's model [183] and a log-normal stock model was used to generate the future scenarios for SMPC.

4.1.3 Transaction Costs

When trading assets on the market, one often suffers the friction due to transaction costs [99]. In mathematical terms, the investor pays a quantity $h_i(t)$ of wealth to change the number of assets in the portfolio from $u_i(t-1)$ at time $t-1$ to $u(t)$ at time t, for each asset i. Such a wealth $h_i(t)$ is taken away from the overall wealth $w(t)$ of the portfolio, so that (4.3) becomes (cf. [312])

$$w(t+1) = (1+r)\left(w(t) - \sum_{i=1}^{n} h_i(t)\right) + \sum_{i=1}^{n} b_i(t)u_i(t). \tag{4.7}$$

In the simplest case, transaction costs $h_i(t)$ are proportional to the traded quantity of stock $|u_i(t) - u_i(t-1)|$,

$$h_i(t) = \epsilon_i |u_i(t) - u_i(t-1)| s_i(t), \tag{4.8}$$

where the fixed quantity ϵ_i depends on commissions on trading asset i, $i = 1, \ldots, n$ (we assume that no costs are applied on transacting the risk-free asset).

Proposition 11. *The variance of the hedging error $e(t) = w(t) - p(t)$ is not affected by transaction costs.*

Proof. Let $\omega(t) = \sum_{i=1}^{n} h_i(t)$ be the total transaction cost paid at time t, which is a function of $u(t)$, $u(t-1)$, and $s(t)$. As $\omega(t)$ clearly does not

depend on $s(t+1)$, the expected value of the hedging error $e(t+1) = w(t+1) - p(t+1)$ taken with respect to $s(t+1)$ is

$$\begin{aligned} \mathrm{E}[w(t+1) - p(t+1)] &= \mathrm{E}[(1+r)w(t) + \\ &\qquad \sum_{i=1}^{n} b_i(t)u_i(t) - p(t+1) - (1+r)\omega(t)] \\ &= \mathrm{E}[w_0(t+1) - p(t+1)] - (1+r)\omega(t), \end{aligned}$$

where $w_0(t+1)$ is the wealth at time $t+1$ in the absence of transaction costs. Therefore, while the expectation $\mathrm{E}[e(t+1)]$ of the hedging error $e(t+1)$ is affected by $\omega(t)$, its variance $\mathrm{Var}[e(t+1)]$ is clearly not, as

$$\begin{aligned} \mathrm{Var}[e(t+1)] &= \mathrm{E}[(e(t+1) - \mathrm{E}[e(t+1)])^2] = \\ &\quad \mathrm{E}[(w_0(t+1) - p(t+1) - (1+r)\omega(t) \\ &\quad -\mathrm{E}[w_0(t+1) - p(t+1)] + (1+r)\omega(t))^2] \\ &= \mathrm{Var}[w_0(t+1) - p(t+1)]. \end{aligned}$$

□

Proposition 11 clearly shows that the minimum variance criterion (4.4) is insensitive to transaction costs and therefore potentially inadequate to handle them.

Constraints on how the quantities $u_i(t)$ are allocated can be additionally imposed. The formulation of optimization problems based on only (4.7) in general permits short-selling, i.e., $u_i(t) < 0$. *Short-selling constraints* can be included as $s_i(t)u_i(t) \geq -S_i^{\text{short}}$ or $\sum_{i=1}^{n} \min(s_i(t)u_i(t), 0) \geq -S^{\text{short}}$ for parameters S_i^{short} and S^{short}, respectively. *Diversification constraints* $s_i(t)u_i(t) \leq S_i^{\max}$ for some constant $S_i^{\max}$, or $s_i(t)u_i(t) \leq \rho_i w(t)$ for some fractional $\rho_i \in (0, 1]$ may be also imposed. Note that all of these constraints are *linear* in the control variables $u_i(t)$, $\forall i = 1, \ldots, n$, a feature that is useful for the hedging formulations discussed in the next sections. Constraints on the *variance* or the *shortfall of risk* ([257]) would lead to convex constraints, although of *second-order cone* type.

4.1.4 SMPC Problem Formulations

For SMPC for dynamic option hedging with transaction costs we use again Algorithm 16, with the only difference that in Step 4 an alternative optimization problem to the least-squares problem is solved. We introduce three possible SMPC formulations to account for transaction costs.

Minimization of Variance and Expectation (QP-Var)

Let $x(t), y(t) \in \mathbb{R}^n$ be two vectors whose i-th components are nonnegative and defined as

$$\begin{aligned} x_i(t) - y_i(t) &= u_i(t) - u_i(t-1), \\ x_i(t) \geq 0,\ y_i(t) &\geq 0,\ \forall t = 0, \ldots, T. \end{aligned} \tag{4.9}$$

Accordingly, the proportional transaction cost $h_i(t)$ for trading a quantity $u_i(t) - u_i(t-1)$ of the i-th asset is $h^i(t) = \epsilon^i |u_i(t) - u_i(t-1)| s_i(t) = \gamma_i(t)(x_i(t) + y_i(t))$, where $\gamma_i(t) \triangleq \epsilon^i s^i(t)$, $i = 1, \ldots, n$. The quantities $x_i(t)$ and $y_i(t)$ can be interpreted, respectively, as the amount of asset i *bought* at time t and the amount of asset i *sold* at time t. We can therefore introduce the new vector $v(t) = \left[\begin{smallmatrix} x(t) \\ y(t) \end{smallmatrix}\right] \in \mathbb{R}^{2n}$ of decision variables and replace $u(t) \in \mathbb{R}^n$ with

$$u(t) = u(t-1) + x(t) - y(t). \tag{4.10}$$

By letting

$$\mathbb{1} \triangleq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^M,\ \gamma(t) \triangleq \begin{bmatrix} \gamma_1(t) \\ \vdots \\ \gamma_n(t) \end{bmatrix},$$

from (4.7) we can express the vector of future hedging errors $e(t+1) = w(t+1) - p(t+1)$ on the M different scenarios as

$$\begin{bmatrix} e^1(t+1) \\ \vdots \\ e^M(t+1) \end{bmatrix} = B(t)u(t) +$$

$$(1+r)\left(w(t) - \gamma^{\mathsf{T}}(t)(x(t)+y(t))\right)\mathbb{1} - \begin{bmatrix} p^1(t+1) \\ \vdots \\ p^M(t+1) \end{bmatrix}$$

$$= B(t)(u(t-1)+x(t)-y(t)) -$$
$$(1+r)\,\mathbb{1}\,\gamma^{\mathsf{T}}(t)(x(t)+y(t)) + D(t)$$
$$= A_v(t)v(t) + B_v(t) - \mathbb{1}\,G_v(t)v(t), \tag{4.11}$$

where

$$B(t) \triangleq \begin{bmatrix} b_1^1(t) & \dots & b_n^1(t) \\ \vdots & & \vdots \\ b_1^M(t) & \dots & b_n^M(t) \end{bmatrix},\ D(t) \triangleq (1+r)\,\mathbb{1}\,w(t) - \begin{bmatrix} p^1(t+1) \\ \vdots \\ p^M(t+1) \end{bmatrix}$$

$$B_v(t) \triangleq B(t)u(t-1) + D(t),\ A_v(t) \triangleq \begin{bmatrix} B^{\mathsf{T}}(t) \\ -B^{\mathsf{T}}(t) \end{bmatrix}^{\mathsf{T}},\ G_v(t) \triangleq (1+r)\begin{bmatrix} \gamma(t) \\ \gamma(t) \end{bmatrix}^{\mathsf{T}}.$$

The hedging error $e(t+1) = w(t+1) - p(t+1)$ has therefore the following empirical expectation

$$\begin{aligned} \mathrm{E}[e(t+1)] &= \pi^{\mathsf{T}}(A_v(t)v(t) + B_v(t) - \mathbb{1}\,G_v(t)v(t)) \\ &= -G_v(t)v(t) + \pi^{\mathsf{T}}(A_v(t)v(t) + B_v(t)), \end{aligned} \tag{4.12}$$

where $\pi^{\mathsf{T}} = [\pi_1\ \dots \pi_M]^{\mathsf{T}} \in \mathbb{R}^M$, $\pi^{\mathsf{T}}\mathbb{1} = 1$. Note that by (4.12) we can rewrite $\mathrm{E}[e(t+1)] = K(t) - H(t)$, where $K(t) = \pi^{\mathsf{T}}(B(t)(x(t)-y(t)) + B_v(t))$ and $H(t) = (1+r)\gamma^{\mathsf{T}}(t)(x(t)+y(t))$. Therefore, $K(t)$ depends on the quantity $x(t) - y(t)$ (i.e., on the net increment $u(t) - u(t-1)$ of the underlying assets hold in portfolio from time $t-1$ to time t) and is independent of $\Lambda(t) = \min(x(t), y(t))$ and of the transaction costs, while $H(t)$ depends on the actual number of transactions executed to rebalance the portfolio at time t, depends on $\Lambda(t)$ and, via $\gamma(t)$, on transaction costs.

By letting i_j be the j-th vector of the canonical basis of $\mathbb{R}^M$, that is $i_j = [\underbrace{0\ \dots\ 0}_{j-1}\ 1\ \underbrace{0\ \dots\ 0}_{M-j}]^{\mathsf{T}}$, and omitting the dependence of t for ease of

notation, we get

$$\begin{aligned}\mathrm{E}[e^2(t+1)] &= \sum_{j=1}^{M}\pi_j\left(i_j^{\mathsf{T}}(A_v v + B_v - \mathbb{1}\,G_v v)\right)^2 = v^{\mathsf{T}}G_v^{\mathsf{T}}G_v v + \\ &(A_v v + B_v)^{\mathsf{T}}\,\mathrm{diag}(\pi)(A_v v + B_v) - 2\pi^{\mathsf{T}}(A_v v + B_v)G_v v \qquad (4.13)\\ \mathrm{E}^2[e(t+1)] &= \left(\sum_{j=1}^{M}\pi_j i_j^{\mathsf{T}}(A_v v + B_v - \mathbb{1}\,G_v v)\right)^2 \\ &= \left(\pi^{\mathsf{T}}(A_v v + B_v) - G_v v\right)^2 \\ &= v^{\mathsf{T}}G_v^{\mathsf{T}}G_v v + (A_v v + B_v)^{\mathsf{T}}\pi\pi^{\mathsf{T}}(A_v v + B_v) - \\ &\qquad 2\pi^{\mathsf{T}}(A_v v + B_v)G_v v. \qquad (4.14)\end{aligned}$$

Hence, the variance of $e(t+1)$ is

$$\begin{aligned}\mathrm{Var}[e(t+1)] &= \mathrm{E}[(e(t+1) - \mathrm{E}[e(t+1)])^2] \\ &= \mathrm{E}[e^2(t+1)] - \mathrm{E}^2[e(t+1)] \qquad (4.15a)\\ &= (A_v(t)v(t) + B_v(t))^{\mathsf{T}}(\mathrm{diag}(\pi) - \pi\pi^{\mathsf{T}})(A_v(t)v(t) + B_v(t)). \qquad (4.15b)\end{aligned}$$

Note that (4.15b) does not depend on $\gamma(t)$, in accordance with Proposition 11, and that $\mathrm{diag}(\pi) - \pi\pi^{\mathsf{T}}$ is a positive semidefinite matrix[1]. Note also that $\mathrm{Var}[e(t+1)]$ does not depend on $x(t) - y(t)$, and therefore on $\Lambda(t)$, that confirms what observed earlier about $\Lambda(t)$ only affecting transaction costs, which are deterministic.

In order to minimize both the variance and the expected value of the one-step ahead hedging error $e(t+1)$ we solve

$$\begin{aligned}\min_{v(t)} \quad & \mathrm{Var}[e(t+1)] + \alpha\mathrm{E}^2[e(t+1)] \qquad (4.16)\\ \text{s.t.} \quad & v(t) \geq 0,\end{aligned}$$

[1]Matrix $\mathrm{diag}(\pi) - \pi\pi^{\mathsf{T}}$ is positive semidefinite by definition: $v^{\mathsf{T}}(\mathrm{diag}(\pi) - \pi\pi^{\mathsf{T}})v = \sum_{i=1}^{M}\pi_i v_i^2 - \left(\sum_{i=1}^{M}\pi_i v_i\right)\left(\sum_{j=1}^{M}\pi_j v_j\right) = \sum_{i=1}^{M}\pi_i\left(v_i^2 - 2v_i\sum_{j=1}^{M}\pi_j v_j + v_i\sum_{j=1}^{M}\pi_j v_j\right) = \left(\sum_{i=1}^{M}\pi_i(v_i^2 - 2v_i\sum_{j=1}^{M}\pi_j v_j)\right) + \left(\sum_{i=1}^{M}\pi_i v_i\right)\left(\sum_{j=1}^{M}\pi_j v_j\right)\left(\sum_{i=1}^{M}\pi_i\right) = \sum_{i=1}^{M}\pi_i\left(v_i - \sum_{j=1}^{M}\pi_j v_j\right)^2 \geq 0, \forall v \in \mathbb{R}^M$, that is the sampled version of (4.15a).

where α is a fixed scalar, $\alpha \geq 0$. Problem (4.16) is a QP problem with $2n$ variables subject to nonnegativity constraints.

The hedging strategy defined by (4.16) might lead to choosing optimal quantities $x_i(t)$ and $y_i(t)$ that are both positive, that is $\Lambda_i(t) \triangleq \min(x_i(t), y_i(t)) > 0$. This amounts to allow the trader to simultaneously buy and sell the same quantity $\Lambda_i(t)$ of asset i at the same trading instant t (cf. [86, p. 290]) or, in alternative, to violate the self-financing condition (4.3), by subtracting the wealth $\Lambda_i(t)\gamma_i(t)$ from the total portfolio wealth and rebalancing $u_i(t) = u_i(t-1) + \bar{x}_i(t) - \bar{y}_i(t)$, where $\bar{x}_i(t) = x_i(t) - \Lambda_i(t)$, $\bar{y}_i(t) = y_i(t) - \Lambda_i(t)$, $\bar{x}_i(t) - \bar{y}_i(t) = x_i(t) - y_i(t)$, and either $\bar{x}_i(t) = 0$ or $\bar{y}_i(t) = 0$. Constraining $\Lambda_i(t) = 0$ would make (4.16) a nonconvex problem, therefore more complicated to solve numerically; however, leaving $\Lambda_i(t)$ unconstrained does not lead to undesired effects from a hedging viewpoint. In fact, having $x_i(t)$ and $y_i(t)$ both positive ($\Lambda_i(t) > 0$) might be a good choice to avoid super-replication without altering the variance of the hedging error. On the other hand, if at optimality $\mathrm{E}[e(t+1)] \leq 0$, that is one is under-replicating the option price at time t, then necessarily $\Lambda_i(t) = 0$, otherwise $\bar{x}_i(t)$, $\bar{y}_i(t)$ would be a solution with the same variance and a lower $\mathrm{E}^2[e(t+1)]$, thus providing a lower value of the objective function in (4.16) than $x(t)$ and $y(t)$.

Note also that one could minimize $\mathrm{Var}[e(t+1)] + \alpha \mathrm{E}[e(t+1)]$ instead of (4.16), therefore not penalizing super-replication. In this setting, either $x_i(t) = 0$ or $y_i(t) = 0$ spontaneously at optimality (that is, $\Lambda_i(t) = 0$ always holds at optimality) because, as observed earlier, a positive quantity $\Lambda_i(t)$ would only increase the term $H(t)$ due to transaction costs without altering $K(t)$ and $\mathrm{Var}[e(t+1)]$.

Remark 8. *An alternative formulation based on mixed-integer quadratic programming, related to the approach of [133] but based on the theory of hybrid dynamical systems [36], that can handle more general transaction costs than proportional costs, can be derived as follows. Piecewise-affine transaction costs as in* (4.8) *can be also handled by introducing binary variables. Let* $x^u(t) \triangleq u(t-1) \in \mathbb{R}^n$ *be the composition of the portfolio immediately before trading at time* t *and introduce auxiliary variables* $\delta_i(t) \in \{0, 1\}$,

$$[\delta_i(t) = 1] \leftrightarrow [u_i(t) - x_i^u(t) \geq 0] \tag{4.17}$$

and $q_i(t) \in \mathbb{R}$

$$q_i(t) = \begin{cases} u_i(t) - x_i^u(t) & \text{if } \delta_i(t) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

By using the so-called "big-M" technique (see cases 1 and 3 for integrating logical constraints in Section 1.3), (4.17) *can be translated into the mixed-integer linear inequalities*

$$u_i(t) - x_i^u(t) \geq -M_i(1 - \delta_i(t)) \tag{4.19a}$$
$$u_i(t) - x_i^u(t) \leq M_i\delta_i(t) - \epsilon \tag{4.19b}$$

and (4.18) *into*

$$q_i(t) \leq u_i(t) - x_i^u(t) + M_i(1 - \delta_i(t)) \tag{4.20a}$$
$$q_i(t) \geq u_i(t) - x_i^u(t) - M_i(1 - \delta_i(t)) \tag{4.20b}$$
$$q_i(t) \leq M_i\delta_i(t) \tag{4.20c}$$
$$q_i(t) \geq -M_i\delta_i(t) \tag{4.20d}$$

where M_i *is an upperbound on* $|u_i(t) - x_i^u(t)|$*, that is the maximum allowed asset reallocation, and* $\epsilon > 0$ *is a small scalar (e.g., the machine precision). Equation* (4.7) *can be therefore interpreted as the evolution of a* hybrid dynamical system, *that is expressed in the following* mixed logical dynamical *(MLD) form [36]*

$$w(t+1) = (1+r)\left(u_0(t) - \sum_{i=1}^{n} q_i(t) - 2(u_i(t) - x_i^u(t))\right) + \sum_{i=1}^{n} s_i(t+1)u_i(t) \tag{4.21a}$$
$$x^u(t+1) = u(t) \tag{4.21b}$$
$$\text{s.t.} \quad (4.19), (4.20) \tag{4.21c}$$

with states $w(t), x^u(t)$*, input* $u(t)$*, auxiliary vector* $\delta(t) = [\delta_1(t) \ \dots \ \delta_n(t)]^T \in \{0,1\}^n$ *of binary variables, and vector* $q(t) = [q_1(t) \ \dots \ q_n(t)]^T \in \mathbb{R}^n$ *of auxiliary continuous variables. Note that, from a system theoretical viewpoint, transaction costs introduce a unit delay* (4.21b) *in the dynamics, due to the additional state variable* $x^u(t)$*. By using the stochastic hybrid dynamical model* (4.21)*, problem* (4.16) *can be recast as a mixed-integer quadratic programming (MIQP) problem (see [36] for details) to be minimized with respect to vector* $u(t) \in \mathbb{R}^n$*, for which very efficient solvers are available [194],*

[168]. See also [133] for a related approach. For options involving a single stock, the number n of assets is usually very small ($n = 1$ or $n = 2$), so that the minimum variance problem with transaction costs can be solved also by enumerating the possible 2^n instances of vector $\delta(t)$ (i.e., in system theoretical terms, by transforming the MLD dynamics (4.21c) *into an equivalent piecewise-affine form [32] and enumerating the "modes" of the resulting PWA dynamics) and by solving the corresponding quadratic programs (QP)* (4.6) *subject to $u_i(t) \geq x_i^u(t)$ if the corresponding $\delta_i(t) = 1$, or $u_i(t) \leq x_i^u(t)$ if $\delta_i(t) = 0$, for all $i = 1, \ldots, n$. While the method of* (4.16) *with proportional transaction cost is in general more efficient from a numerical viewpoint, in that it completely avoids introducing integer variables, the MIQP method is more general, for example it can be easily extended to handle transaction costs of the form $h_i(u_i(t) - u_i(t-1)) = \min(c_0, \epsilon^i s_i(t)|u_i(t) - u_i(t-1)|)$, where c_0 is a given minimum fixed cost to be paid in each transaction.*

Before stating the second main SMPC formulation, we motivate (4.16) differently. Starting from a minimum-variance objective, that alone is not suitable to account for transaction costs according to Proposition 11, we extend it by a probabilistic *chance-constraint* [78] as follows

$$\min_{v(t)\geq 0} \quad \mathrm{Var}[e(t+1)] \tag{4.22a}$$

$$\text{s.t.} \quad p(e(t+1) \leq e_{\text{low}}) \leq \eta, \tag{4.22b}$$

where we treat $e(t+1)$ as a random variable defined on some probability space and parameterized by $v(t)$, where $p(e_{t+1} \leq e_{\text{low}})$ denotes the probability of event $e_{t+1} \leq e_{\text{low}}$, and where e_{low} and $\eta \in [0, 1]$ are two parameters. Then, under the assumption that $e(t+1)$ is Gaussian distributed, $e(t+1) \sim \mathcal{N}(\mu(t+1), \sigma(t+1))$, we obtain $p(e(t+1) \leq e_{\text{low}}) = N\left(\frac{e_{\text{low}} - \mu(t+1)}{\sqrt{\sigma(t+1)}}\right)$, where $N(\cdot)$ denotes the cumulative distribution function (CDF) of the standard normal distribution. Distinguishing between two cases, $\eta \in [0.5, 1]$ and $\eta \in [0, 0.5)$, we obtain for (4.22b), $e_{\text{low}} - \mu(t+1) \leq N^{-1}(\eta)\sqrt{\sigma(t+1)}$ and $\mu(t+1) - e_{\text{low}} \geq N^{-1}(1-\eta)\sqrt{\sigma(t+1)}$, re-

spectively. Thus, for the first case we can formulate equivalently

$$\min_{v(t)\geq 0} \quad \mathrm{Var}[e(t+1)] \tag{4.23a}$$

$$\text{s.t.} \quad (e_{\text{low}} - \mu(t+1))^2 \leq \left(N^{-1}(\eta)\right)^2 \sigma(t+1), \tag{4.23b}$$

$$(e_{\text{low}} - \mu(t+1)) \geq 0, \tag{4.23c}$$

and similarly for the second case. Here a remark needs to be made. Suppose we parameterize $e(t+1)$ according to (4.12) and make the additional (simplistic) assumption that $b_i(t) = s_i(t+1) - (1+r)s_i(t)$ is Gaussian distributed, then $e(t+1)$ is likewise Gaussian (by linearity of (4.12)) and thus fits above framework. However, then (4.23) is in general *not* a quadratically constrained *convex* program. This is since the Hessian of (4.23b), which is here considered as a second-order condition for convexity, is in general not positive semidefinite. Instead, we consider the softened version of (4.23) as $\min_{v(t)\geq 0}\{\mathrm{Var}[e(t+1)]+...$
$\lambda\left((e_{\text{low}} - \mu(t+1))^2 \leq \left(N^{-1}(\eta)\right)^2 \sigma(t+1)\right) + \xi(\mu(t+1) - e_{\text{low}})\}$ with Lagrangian multipliers $\lambda, \xi \in \mathbb{R}$. This resembles (4.16) for $e_{\text{low}} = 0$ and $\xi = 0$. In fact, after a scaling step (not affecting the minimizer), it translates to

$$\min_{v(t)\geq 0} \quad \mathrm{Var}[e(t+1)] + \tfrac{\lambda}{1-\lambda(N^{-1}(\eta))^2}\mathrm{E}^2[e(t+1)], \tag{4.24}$$

with $0 \leq \lambda < \frac{1}{(N^{-1}(\eta))^2} \in [1, 4]$ (derived from the equivalent $\alpha \geq 0$ in (4.16)) for $\eta \in [0.5, 1]$. Similarly, the case for $\eta \in [0, 0.5]$ can be obtained with $\alpha \geq 0$ in (4.16) being represented by $\frac{-\lambda}{1+\lambda(N^{-1}(1-\eta))^2}$, and thus $0 \geq \lambda > -\frac{1}{(N^{-1}(1-\eta))^2} \in [-1, -4]$ to have $\alpha \geq 0$.

To summarize, we motivated (4.16) starting from a minimum-variance objective and added chance constraint (4.22b) to account for transaction costs. It is stressed that only under the assumption of $e(t+1)$ following a Gaussian distribution and a relaxation of (4.22b), a (loose) relation to (4.16) could be established. Note that the chance-constraint formulation (4.22) permits to formulate *general* probabilistic constraints (without making the assumption of a Gaussian distribution of $e(t+1)$). Then, following [292], *convex approximations* of general chance constraints (4.22b)

can be formulated by means of different *generating functions* that place different penalties on how tight the original chance constraints are approximated.

Finally, let us also draw a relation between the β-VaR (Value at Risk) and the corresponding chance-constraint optimization problem formulation, as well as state the associated LP in our scenario-based SMPC framework. Further below, the β-VaR is used to derive the β-CVaR (Conditional Value at Risk) and defined by a minimization problem after the definition of a loss function. For contrast, we here directly work with $e(t+1)$ and consider a more intuitive maximization problem formulation. We here define the β-VaR as e_{low} being the solution of the *chance-constraint* optimization problem as follows:

$$\max_{v(t)\geq 0, e_{\text{low}}} \quad e_{\text{low}} \tag{4.25a}$$

$$\text{s.t.} \quad p(e_{t+1} \leq e_{\text{low}}) \leq 1-\beta, \tag{4.25b}$$

where β is a parameter, typically $\beta = 90\%,\ 95\%$ or 99%. For our *scenario-based* approach of (4.11), we can approximate (4.25) as

$$\max_{v(t)\geq 0, e_{\text{low}}} \quad e_{\text{low}} \tag{4.26a}$$

$$\text{s.t.} \quad \| \max(e_{\text{low}} \cdot \mathbb{1} - \left[e^1(t+1),\ \ldots,\ e^M(t+1)\right], \ldots \ldots \mathbb{0}) \|_0 \leq (1-\beta)M, \tag{4.26b}$$

where $\mathbb{0}$ indicates a vector of zeros, $\|\cdot\|_0$ denotes the ℓ_0-norm and the $\max$-operator is acting element-wise. After convexifying constraint (4.26b) by substituting the ℓ_0- with the ℓ_1-norm and exploiting the natural non-negativity of the formulation permitting us to drop absolute values, we obtain

$$\max_{v(t)\geq 0, e_{\text{low}}} \quad e_{\text{low}} \tag{4.27a}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} \max(e_{\text{low}} - e^i(t+1),\ 0) \leq (1-\beta)M. \tag{4.27b}$$

Ultimately, from (4.27) we obtain the final LP-formulation:

$$\max_{v(t)\geq 0, e_{\text{low}}, \{y_i\}_{i=1}^{M}} \quad e_{\text{low}} \tag{4.28a}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} y_i \leq (1-\beta)M, \tag{4.28b}$$

$$y_i \geq e_{\text{low}} - e^i(t+1), \tag{4.28c}$$

$$y_i \geq 0. \tag{4.28d}$$

As motivated by [324], the CVaR can be considered to be a more consistent measure of risk than VaR. Therefore, in the evaluation Section 4.1.6, we only consider the CVaR-based SMPC formulation introduced next rather than (4.28).

For very recent work on chance-constraints with applications to portfolio optimization, see also [348] and [334].

Minimization of Conditional Value at Risk (LP-CVaR)

A drawback of the QP formulation (4.16) is that it requires calibrating the scalar α that achieves the best tradeoff between variance (=risk) and expectation (=lack of hedging accuracy due to transaction costs). *Conditional Value at Risk* (CVaR) can be used as an alternative performance measure to penalize the hedging error $e(t+1)$, and is defined as follows.

Let $f(u,s) : \mathbb{R}^{n+k} \to \mathbb{R}$ be a loss function associated with the decision vector $u \in \mathbb{R}^n$ and with the random vector $s \in \mathbb{R}^k$. In our case $u = u(t)$, $s = m(t+1)$, $f(u,s) = |e(t+1)|$ (in case super-replication of the option price is not penalized, $f(u,s) = -e(t+1)$). Let $p(s)$ be the probability density function of s. With respect to a given probability β, $0 \leq \beta \leq 1$, the β-VaR (Value at Risk) is defined as the lowest value ℓ, such that, with probability β, the loss will not exceed ℓ. The number β is a fixed value, typically $\beta = 90\%$, 95%, or 99%. The main drawback of VaR is that the amount of loss occurring with probability $(1-\beta)$ is not taken into account directly. To avoid this, β-CVaR was introduced, that is the conditional expectation of the loss function above ℓ, quantifying what the *average loss*

is when one loses *more than* ℓ, with probability $1-\beta$ [324]. The probability of $f(u,s)$ not exceeding the threshold ℓ is

$$\psi(u,\ell) = \int_{f(u,s)\leq\ell} p(s)ds.$$

The β-VaR and the β-CVaR are defined, respectively, as

$$\ell_\beta(u) = \min\{\ell \in \mathbb{R} :\ \psi(u,\ell) \geq \beta\}$$

and

$$\phi_\beta(u) = \frac{1}{1-\beta}\int_{f(u,s)\geq\ell_\beta(u)} f(u,s)p(s)ds.$$

In [324] it is shown that the β-CVaR of the loss associated with any u can be determined by the formula

$$\phi_\beta(u) = \min_{\ell\in\mathbb{R}} F_\beta(u,\ell),$$

where

$$F_\beta(u,\ell) = \ell + \frac{1}{1-\beta}\int_{s\in\mathbb{R}^m} [f(u,s)-\ell]^+ p(s)ds \tag{4.29}$$

and $[\cdot]^+$ denote the positive part of its argument, $[f]^+ = \max(f,0)$. The integral in (4.29) can be approximated by sampling the distribution of s, according to the density function $p(s)$. If the sampling generates a collection of M vectors $s^1,\ \ldots,\ s^M$, each of which has probability π_j of occurring, $j = 1,\ldots,M$, then the corresponding approximation $\tilde{F}_\beta(u,\ell)$ is

$$\tilde{F}_\beta(u,\ell) = \ell + \frac{1}{1-\beta}\sum_{j=1}^{M} \pi_j [f(u,s^j)-\ell]^+.$$

Finally, we use the CVaR to formulate the SMPC problem for dynamic

hedging:

$$\begin{aligned}
\min_{v(t),\ell(t),\{z_j(t)\}_{j=1}^M} \quad & \ell(t) + \frac{1}{1-\beta}\sum_{j=1}^{M}\pi_j z_j(t) && \text{(4.30a)}\\
\text{s.t.} \quad & z_j(t) \geq w^j(t+1) - p^j(t+1) - \ell && \text{(4.30b)}\\
& z_j(t) \geq -w^j(t+1) + p^j(t+1) - \ell && \text{(4.30c)}\\
& z_j(t) \geq 0 && \text{(4.30d)}\\
& \quad j = 1, \ldots, M \\
& v(t) \geq 0 && \text{(4.30e)}
\end{aligned}$$

for the given fixed value of β, where $w^j(t+1) - p^j(t+1)$ is given by (4.11). Problem (4.30) is a Linear Programming (LP) problem with $M + 2n + 1$ variables and $3M + 2n$ constraints. Note that by removing constraint (4.30b) one does not penalize super-replication of the option price, as the loss function becomes $\max(-e(t+1), 0)$.

Minimization of Worst-case Error (LP-MinMax)

A simpler approach than CVaR is to penalize the worst-case loss over the set of M generated scenarios, that is the largest absolute value $|e(t+1)|$ of the hedging error. The resulting formulation is the following LP problem

$$\begin{aligned}
\min_{v(t),\ell(t)} \quad & \ell(t) && \text{(4.31a)}\\
\text{s.t.} \quad & \ell(t) \geq w^j(t+1) - p^j(t+1) && \text{(4.31b)}\\
& \ell(t) \geq -w^j(t+1) + p^j(t+1) && \text{(4.31c)}\\
& \quad j = 1, \ldots, M \\
& \ell(t) \geq 0 && \text{(4.31d)}\\
& v(t) \geq 0, && \text{(4.31e)}
\end{aligned}$$

where $w^j(t+1) - p^j(t+1)$ is given by (4.11). Note that the LP (4.31) is simpler than (4.30) as it only involves $2n + 1$ variables and $2(M + n) + 1$

constraints (they are identical for $M = 1$). In contrast, it is clear that the minmax formulation (4.31) does not exploit the available information about the probability distribution of the stochastic variables that affect the evolution of the portfolio.

Finally, alternative performance measures to penalize the hedging error $e(t+1)$ are possible, such as the average of the maximum *shortfall*, with shortfall for scenario j defined as $\max(-(w^j(t+1) - p^j(t+1)), 0)$. In addition, terms penalizing transactions may be added to the objective functions, and, e.g., transaction-rate constraints may be introduced.

4.1.5 Scenario Generation

The closed-loop performance of SMPC heavily depends on the way the scenarios of both $s^j(t+1)$ and $p^j(t+1)$, $j = 1, \ldots, M$, are generated.

Stock Models

We propose three scenario generation methods for stock prices:

1. logn. The most widely used model to describe the dynamics of stock prices is the log-normal (logn) model. Its discrete-time form is

$$s_i(t+1) = s_i(t) e^{(\mu_i - \frac{1}{2}\sigma_i^2)T_s + \sigma_i \sqrt{T_s}\eta_i(t)}, \tag{4.32}$$

with T_s the sampling interval (e.g., 1 day), $\eta_i(t) \sim \mathcal{N}(0,1)$, $\forall i = 1, \ldots, n$. Parameters μ_i and σ_i must be estimated from data; typically as the *maximum likelihood* (ML) estimates from $\mathcal{T}+1$ past stock prices using $\left\{ \ln\left(\frac{s_i(t-\mathcal{T}+1)}{s_i(t-\mathcal{T})}\right), \ldots, \ln\left(\frac{s_i(t)}{s_i(t-1)}\right) \right\}$ and exploiting the Gaussian distribution of $\eta_i(t)$. We tested three methods. First, after ML-identification on the training data, we maintained estimates μ_i and σ_i constant throughout the option's life. Second, we recursively re-estimated them using ML on a time-shifted data set (up until the current hedging date) of constant window length $\mathcal{T}+1$. Third, after initialization at $t = 0$ using the ML-estimate, we applied a *discrete Extended Kalman Filter* (dEKF) to adapt online. While for artificially generated toy examples with underlying price

$s(t)$ following a logn model, dEKF-estimates converged to the true parameters, for real-world data it was not the case (real-world data does not follow the log-normal stock model). An additional disadvantage of the dEKF-solution is the difficulty to select suitable tuning parameters (trading-off model predictions and actual measurements). The recursive ML-estimation performed overall best and is our preferred method when using the logn method. After identification and given $s_i(t),\ \forall i = 1, \ldots, n$, at current hedging date t, the M scenarios are generated by drawing $\eta_i^j(t) \sim \mathcal{N}(0,1)$ before evaluating (4.32) to obtain $s_i^j(t+1)$ and $\pi_i^j = \frac{1}{M},\ \forall j = 1, \ldots, M$.

2. SVR. This second model for stock-price predictions is based on *support vector regression* (SVR) using Vapniks's ϵ-insensitive loss function for one-dimensional outputs, see [342]. The guiding motivation is to derive a parametric *nonlinear* fit to past stock data, with input signal being the time instances $\{t - \mathcal{T}_{\text{SVR}}, \ldots, t\}$, and the output signal the corresponding stock prices $\{s_i(t - \mathcal{T}_{\text{SVR}}), \ldots, s_i(t)\}$. SVR can generate excellent nonlinear fits to past stock data. This motivated us to use the identified model for a one-step ahead prediction. The prediction model has the form $\hat{s}_i(t+1) = W^{\mathsf{T}}\varphi(t+1)+q$ with parameters $W \in \mathbb{R}^{n_f \times 1}$, where n_f denotes a high-dimensional feature space dimension, $q \in \mathbb{R}$ and $\varphi(\cdot) : \mathbb{R} \to \mathbb{R}^{n_f \times 1}$. According to *Mercer's Theorem*, $\varphi(t)^{\mathsf{T}}\varphi(\tilde{t}) = K(t, \tilde{t})$ with symmetric and positive definite kernel function, e.g., for the radial basis function (RBF) kernel: $K(t, \tilde{t}) = e^{(-\|t-\tilde{t}\|)/\sigma_{\text{RBF}}^2}$. The tuning parameters of the method are the positive scalars $\mathcal{T}_{\text{SVR}}$, σ_{RBF}, C_{SVR} and ϵ_{SVR}. Then, the optimization problem for SVR reads:

$$\min_{\substack{W,q,\xi_\tau,\xi_\tau^* \\ \tau=0,\ldots,\mathcal{T}_{\text{SVR}}}} \quad \frac{1}{2}\|W\|_2^2 + C_{\text{SVR}}\mathbb{1}^{\mathsf{T}}\xi_\tau + C_{\text{SVR}}\mathbb{1}^{\mathsf{T}}\xi_\tau^* \tag{4.33a}$$

$$\text{s.t.} \quad \text{given data: } \{t-\tau, s_i(t-\tau)\}_{\tau=0}^{\mathcal{T}_{\text{SVR}}}, \tag{4.33b}$$

$$\xi_\tau, \xi_\tau^* \geq 0, \tag{4.33c}$$

$$s_i(t-\tau) - W^{\mathsf{T}}\varphi(t-\tau) - q \leq \epsilon_{\text{SVR}} + \xi_\tau, \tag{4.33d}$$

$$-s_i(t-\tau) + W^{\mathsf{T}}\varphi(t-\tau) + q \leq \epsilon_{\text{SVR}} + \xi_\tau^*, \tag{4.33e}$$

where $\mathbb{1}$ denotes a column-vector of ones. We solve Problem (4.33) in the standard way by first formulating and solving its *dual* problem, which is a quadratic program (QP), and then determining $q \in \mathbb{R}$ via the Karush-Kuhn-Tucker (KKT) conditions. Thus, for the solution of (4.33), only a QP-solver is required. The step-ahead stock price prediction is then conducted using the dual optimization variables, training input data and applying Mercer's Theorem. We then generate the M scenarios from

$$s_i^j(t+1) = \hat{s}_i(t+1) + 2\delta_{\mathsf{SVR}}\eta_i^j(t),\ \eta_i^j(t) \sim \mathcal{N}(0,1),\ j = 1,\ldots,M \tag{4.34}$$

for all $i = 1,\ldots,n$ with $\delta_{\mathsf{SVR}} = \frac{1}{\mathcal{T}_{\mathsf{SVR}}}\sum_{\tau=1}^{\mathcal{T}_{\mathsf{SVR}}} |s_i(0+1-\tau) - s(0-\tau)|$, i.e., identified from the offline training data set. The coefficient 2 in (4.34) was determined from closed-loop experiments. We found that a relatively high value was required for improved robustness, see also Section 4.1.6 for a related discussion.

3. pert. The proposed third model for stock price predictions takes the current stock price as the mean estimate (*Martingale process*) and generates scenarios by adding white perturbation noise, i.e.,

$$s_i^j(t+1) = s_i(t) + \sigma_{\mathsf{pert}}\eta_i^j(t),\ \eta_i^j(t) \sim \mathcal{N}(0,1),\ j = 1,\ldots,M \tag{4.35}$$

for all $i = 1,\ldots,n$.

For final closed-loop simulations, we considered one year (252 trading days) of past real-world stock prices, which we partitioned into $\mathcal{T} = 125$ days of training data for initialization of μ and σ estimates (logn model). The coefficient $\sigma_{\mathsf{pert}} = 0.3$ for the pert model was determined experimentally from both artificially generated and real-world data in closed-loop hedging experiments. We likewise determined $\sigma_{\mathsf{RBF}} = 100$, $C_{\mathsf{SVR}} = 1$ and $\epsilon_{\mathsf{SVR}} = 0.01$. For the SVR-model, an interesting finding was that the very short time period $\mathcal{T}_{\mathsf{SVR}} = 10$ in combination with relatively large perturbation variance $(2\delta_{\mathsf{SVR}})^2$ yielded the best closed-loop hedging results. Even if the presented SVR-scheme permits in practice arbitrarily accurate nonlinear fits to *past* stock price data, the correspondingly identified model does *not* enable correct one-step ahead

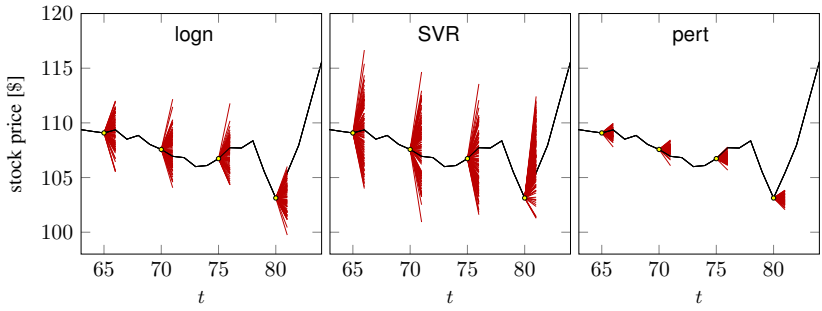


Figure 67: Comparison of the identified three scenario generation methods for the prediction of $s(t+1)$. The black line indicates the true stock price. The red lines indicate the scenarios $s^j(t+1),\ j=1,\ldots,M=100$ at times $t \in \{65, 70, 75, 80\}$.

stock price *predictions* with the same accuracy (also not even by sign). For the three methods, the average computation time for the generation of all of $s^j(t+1),\ j=1,\ldots,M=100$ at each trading date was 0.14ms, 0.4ms (including the time for building and solving of the dual QP) and 0.018ms. As expected, the SVR solution requires by far the most computations. Figure 67 quantitatively visualizes the three final scenario generation methods for the prediction of $s(t+1)$.

Option-pricing Engine

An option-pricing engine is needed at Step 3 of Algorithm 16 to estimate future option prices $p^j(t+1) = (1+r)^{-(T-(t+1))}\mathrm{E}[p^j(T)|s^j(t+1)],\ \forall j = 1,\ldots,M$. By employing Monte Carlo (MC) simulations and the log-normal stock model, estimates for a European call option can be computed from

$$s^j_{i,k}(t+l) = s^j_{i,k}(t+l-1)e^{(\mu_i - \frac{\sigma_i^2}{2})T_s + \sigma_i\sqrt{T_s}\eta_i(t+l-1)}, \tag{4.36}$$

$$\mathrm{E}[p^j(T)|s^j(t+1)] = \frac{1}{N_{\text{sim}}}\sum_{k=1}^{N_{\text{sim}}} \max(s^j_{i,k}(T) - K, 0), \tag{4.37}$$

with $i=1$, $\eta_i(t+l-1) \sim \mathcal{N}(0,1)$, $l=2,\ldots,T-t$, $s^j_{i,k}(t+1) = s^j_i(t+1),\ \forall k = 1,\ldots,N_{\text{sim}}$. Note that $i=1$ since there is one stock

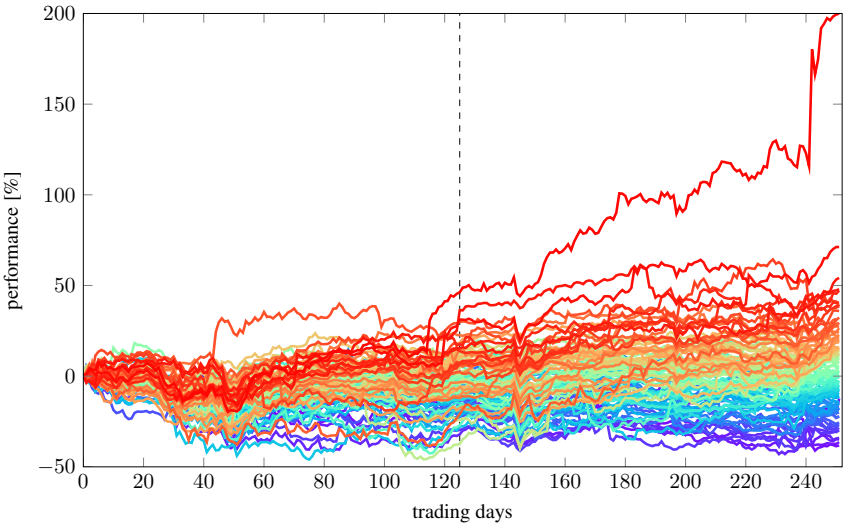


Figure 68: Normalized one-year evolution of all 105 stocks held in NASDAQ-100 between November 27, 2015, and November 25, 2016. The 252 trading days are partitioned by the black-dashed vertical line into training and option evaluation data, respectively. Thus, $t = 0$ is initialized at trading day 125.

underlying both a European call and a Barrier option. Even for a replicating portfolio holding $n - 1$ other assets, the option underlying stock shall always be identified with $i = 1$. Thus, starting from $s_1^j(t+1)$, additional N_{sim} scenarios (e.g., 100) up until expiration date T are generated. For the *path-dependent* Barrier option (4.37) is replaced according to (4.2).

As an alternative, we tested the simpler following pricing scheme for European call option prices

$$p^j(t+1) = (1+r)^{-(T-(t+1))} \max(s_i^j(t+1) - K, 0), \tag{4.38}$$

$i = 1,\ j = 1, \ldots, M$, and similarly for the barrier option. Thus, in comparison to the first method, we implicitly assumed $s_{1,k}^j(t+l) = s_{1,k}^j(t+1),\ \forall l = 2, \ldots, T-t,\ k = 1, \ldots, N_{\text{sim}}$.

In fact, in addition to being much faster[2], the second pricing engine

[2]The average computation time was 0.6s and 6.6e-5s, respectively, for the generation of

(4.38) yielded significantly better and more consistent closed-loop hedging results. Intuitively, this has the following reason. Real-world data do not follow a log-normal stock model. Thus, (4.36) can only very crudely predict $s_i(T)$. This is especially the case for a large difference between the current hedging date t and the expiration date T.

4.1.6 Hedging Results

We test the three SMPC formulations for dynamic hedging defined, respectively, by (4.16), (4.30), and (4.31) on a European plain vanilla call option and on a barrier option with scenarios of stock and option prices generated according to Section 4.1.5. For the QP-Var approach we select $\alpha = 0.25$, as it was calibrated in [42] using simulations of a log-normal stock model and assuming real market generating prices according to the same model (idealized nominal case). For the LP-CVaR approach we use $\beta = 95\%$ in (4.30).

For the solution of the QPs and LPs, we employ the domain-specific language CVXPY for optimization embedded in Python, see [92]. For completeness, QP- and LP-solution times are reported. Since they are in the ms-range, the typical QP- and LP-complexities are not limiting factors for daily rebalancing. For more frequent rebalancing, however, they may become relevant. Note that the original Δ-hedging theory ([49]) is based on a *continuous-time* rebalancing assumption. Numerical experiments were run in Python 2.7 on a laptop running Ubuntu 14.04 equipped with an Intel Core i7 CPU @2.80GHz×8, 15.6GB of memory.

In this section we present results for real-world price data of all 105 stocks held in the NASDAQ-100 between November 27, 2015, and November 25, 2016. Data were obtained from `finance.yahoo.com`[3], see Figure 68 for visualization. We initialize $t = 0$ at trading day 125 (May 27, 2016). The option was simulated to expire after $T = 127$ rebalancing intervals (on November 25, 2016 with daily rebalancing). We assume

$p^j(t+1),\ j = 1, \ldots, M = 100$. Computation time is relevant since it permits to increase the number of scenarios M.

[3]The stock split (ratio two for one) that took place on December 2, 2015 for Ctrip.com International Ltd. was not accounted for in the retrieved data.

proportional transaction costs of 1% and an effective annual risk-free rate $r_a = 1\%$. For each stock $s_i(t),\ i = 1, \ldots, 105$, we assume a European call option with strike price $K = s_i(0)$ and initialize $w(0) = 0.01 s_i(0)$. For the UP-AND-OUT option the barrier is set as $s_u = 1.1 s_i(0)$. In all cases the replicating portfolio is composed by the underlying stock and a cash position (a set-up similar to common Δ-hedging). We therefore drop subscripts i in the following.

A standard option contract typically covers 100 shares. Thus, $u(t) = 1$ implies a portfolio such that at the end of the rebalancing interval 100 shares of the underlying asset are held. Throughout the plots of this section, the average one-step ahead predicted option price is denoted by $\bar{p}(t+1) = \frac{1}{M} \sum_{j=1}^{M} p^j(t+1)$, the true option price by $p^\star(t) = (1 + r)^{-(T-t)} \max(s(T) - K, 0)$ for a European call option and similarly for the path-dependent barrier option. For both options, we initialize $w(0) = 0.01 s(0)$. This simplistic wealth initialization is used for the reason that it permits good evaluation of tracking capabilities of the controllers. Since the second option-pricing engine of Section 4.1.5 is our preferred choice, we obtain $\bar{p}(0) = 0$. This is because in experiments, as outlined above, we initialize $K = s(0)$. Since stock prices typically cost much less than 1000$, our $w(0)$-choice implies a small initial estimated hedging error $w(0) - \bar{p}(0)$. Tracking capabilities of the controller can then be evaluated when proceeding with $t = 0, \ldots, T$. For visualization, see Figures 71 and 72 for small t. Note that we do not initialize $w(0)$ (and accordingly $w(0) - \bar{p}(0)$) *uniformly* for all NASDAQ-100 stocks. This is not done as a means of testing robustness. In practice, $w(0)$ is initialized ideally as the true option price (which is unknown in a causal setting at $t = 0$) plus a premium. In general, it may be very difficult to select an appropriate $w(0)$, especially in the presence of transaction costs. Above presented methods offer a practical tool to financial institutions to *simulate* the effect of different initial prices $w(0)$ and choose a proper one.

The objective of the reported simulations is to understand what is the most suitable SMPC algorithm and scenario generation scheme, how they perform in comparison to Δ-hedging, and how perfect one-step ahead knowledge affect results. For the last issue, we will assume that at

Table 10: Results for a European call option without knowledge of the exact stock price $s(t+1)$ at time t. The CPU-time (ms) for building and solving of the QP, LPs and for Δ-hedging is given by $\bar{\tau}$. For LP-CVaR and $M = 1000$ no solution could be returned for some of the stocks considered.

Controller	M	$\mathbf{E}[e(T)]$ logn/SVR/pert	$\min(\mathbf{e}(\mathbf{T}))$ logn/SVR/pert	$\mathbf{Var}[e(T)]$ logn/SVR/pert	$\bar{\tau}$, ms
QP-Var	100	$-9.6/-19.3/-7.2$	$-124.7/-139.0/-193.2$	216.7/634.3/382.5	3.3
LP-CVaR	100	$-14.7/-14.0/-7.1$	$-134.0/-117.0/-45.4$	387.9/306.9/51.0	4.3
LP-MinMax	100	$-16.2/-16.5/-9.1$	$-150.6/-131.5/-94.8$	458.8/459.2/132.3	3.2
QP-Var	1000	$-9.2/-17.4/-7.1$	$-137.9/-146.0/-165.1$	226.5/558.2/281.7	12.7
LP-CVaR	1000	$-/-/-$	$-/-/-$	$-/-/-$	–
LP-MinMax	1000	$-15.7/-13.9/-10.6$	$-179.3/-89.8/-189.3$	522.0/309.5/361.5	10.6
Δ-hedging		-12.0	-114.6	282.4	$6e-2$

Abbreviations: LP, linear program; LP-CVaR, linear program conditional value at risk; QP-Var, quadratic program variance; SVR, support vector regression.

Table 11: Results for a European call option in the *noncausal* case with perfect knowledge of exact stock price $s(t+1)$ at time t. In accordance with the scenario generation of (4.40), we set $M = 100$.

Controller	$\mathbf{E}[e(T)]$	$\min(e(\mathbf{T}))$	$\mathbf{Var}[e(T)]$	τ, ms
QP-Var	3.3	-0.6	109.8	3.3
LP-CVaR	3.4	-0.6	123.8	3.8
LP-MinMax	3.3	-0.9	123.8	3.0
Δ-hedging	-2.8	-45.7	37.9	$6e-2$

Abbreviations: LP, linear program; LP-CVaR, linear program conditional value at risk; QP-Var, quadratic program variance.

a given time t we know $s(t+1)$ (but not $s(t+2),\ s(t+3),\dots$). Finally, we want to assess in general whether SMPC can have significant practical application for dynamic option-hedging with transaction costs.

European Call Option

We first test the SMPC algorithm on a European call option. Table 10 summarizes the expected and most negative final hedging error $e(T)$ and its variance for the considered stock data, see also Figure 69. A zero (or even positive) $\min(e(T))$ is desired, as it indicates the wealth shortfall at expiration. For Δ-hedging we employed the analytical hedging formula

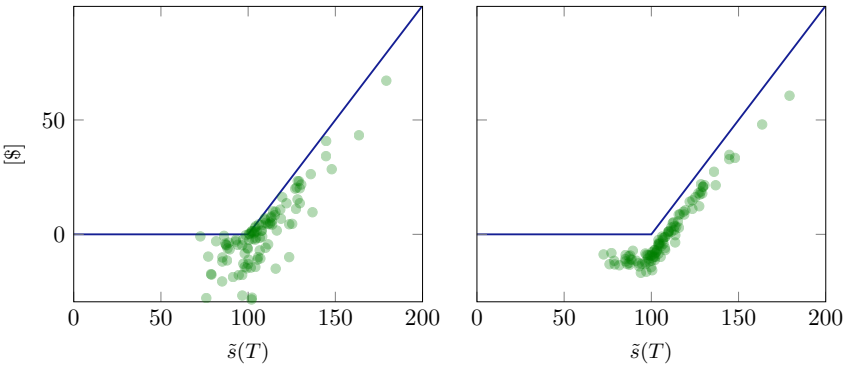


Figure 69: Results for a European call option in the causal case. The normalized portfolio wealth $\tilde{w}(T) = 100w(T)/K$ and corresponding $\tilde{s}(T) = 100s(T)/K$ for all 105 stocks (green dots) is compared to the normalized payoff $\tilde{p}(T) = \max(\tilde{s}(T) - 100, 0)$ (solid blue) at expiration date T. (Left) Solution for LP-CVaR, $M = 100$ and pert, see the second row from the top in Table 10. (Right) Solution for Δ-hedging, see the last row from the top in Table 10.

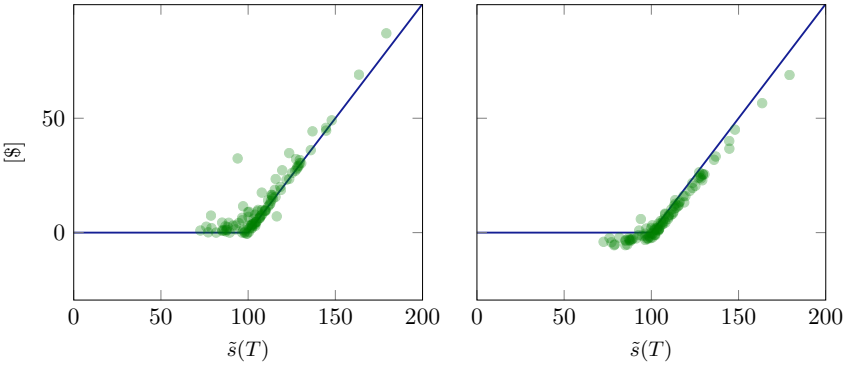


Figure 70: Results for a European call option in the noncausal setting. The normalized portfolio wealth $\tilde{w}(T) = 100w(T)/K$ and corresponding $\tilde{s}(T) = 100s(T)/K$ for all 105 stocks (green dots) is compared to the normalized payoff $\tilde{p}(T) = \max(\tilde{s}(T) - 100, 0)$ (solid blue) at expiration date T. (Left) Solution for LP-CVaR, $M = 100$ and pert, see the second row from the top in Table 11. (Right) Solution for Δ-hedging, see the last row from the top in Table 11.

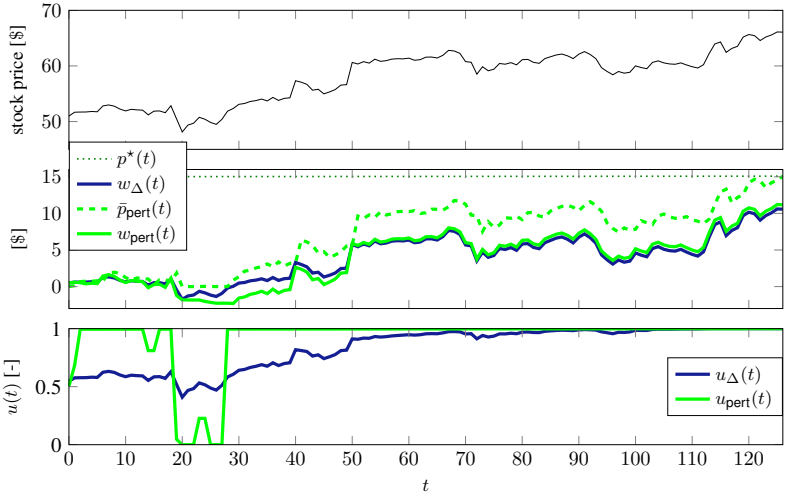


Figure 71: Causal setting. Comparison of Δ-hedging and LP-CVaR for a European Call option. The underlying stock price (top frame) is of Microchip Technology Inc. (May 27 until November 25, 2016).

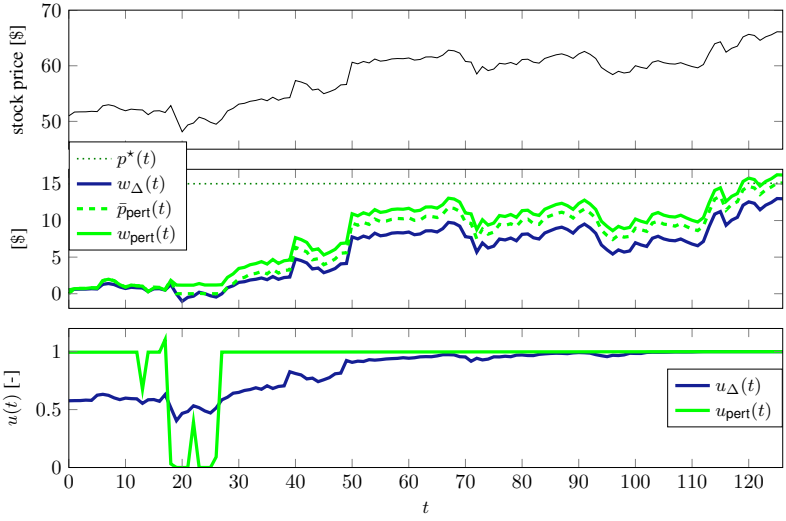


Figure 72: Noncausal setting. Comparison of Δ-hedging and LP-CVaR for a European Call option. The underlying stock price (top frame) is of Microchip Technology Inc. (May 27 until November 25, 2016).

for $u(t)$ from [49],

$$u(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{d_1(t)} e^{-\frac{\xi^2}{2}} d\xi = N\left(d_1(t)\right), \tag{4.39}$$

with $d_1(t) = \left(\ln\left(\frac{s(t)}{K}\right) + \left(r_c + \frac{\sigma^2}{2}^2\right)(T-t)\right) / \left(\sigma\sqrt{T-t}\right)$, and whereby $r_c = \frac{\ln(1+r_a)}{T}$ is the continuously compounded interest rate (and r_a the effective *annual* risk-free rate), and σ recursively estimated at each t as the maximum likelihood as done for the logn-scenario generation method, and where $N(\cdot)$ denotes the cumulative distribution function (CDF) of the standard normal distribution. Δ-hedging results in Table 10 refer to above method. We also tested the control law $u_t = \frac{p(t)-p(t-1)}{s(t)-s(t-1)}$ employing only known price data at t by computing $p(t) = (1+r)^{-(T-t)} \max(s(t) - K, 0)$ and similarly $p(t-1)$. In addition, we tested generating M scenarios $s^j(t+1),\ \forall j$, before computing an average and the derivative approximation to account for the *step-ahead* nature of $\frac{\partial p(t+1)}{\partial s(t+1)}$. This, however, did not yield improvements. Figure 71 illustrates typical rebalancing trajectories for Δ-hedging and for a SMPC-based algorithm, here, using LP-CVaR and pert for scenario generation. Characteristic for Δ-hedging is the rebalancing at almost every sampling time until reaching saturation, see (4.39). The control command associated with SMPC is much less jagged, often of step-wise nature and displaying variations more sparsely. These properties could be observed in multiple experiments.

Secondly, we tested the SMPC in a *noncausal* setting: at every time t we assumed a perfect *one-step* ahead knowledge of price $s(t+1)$. Consequently, we replaced the three stock price scenario generation schemes from Section 4.1.5, and instead used

$$s^j(t+1) = s(t+1) + \sigma_{\text{pert}}\eta^j(t),\ \eta^j(t) \sim \mathcal{N}(0,1),\ j = 1, \ldots, M, \tag{4.40}$$

which is identical to (4.35) except that $s(t+1)$ replaces $s(t)$ as the mean. The perturbation parameter σ_{pert} is set to 0.3 as for the causal setting. The reason for maintaining perturbation noise is to robustify the SMPC algorithm and is discussed in greater detail in the next section. For Δ-hedging in the noncausal setting, we still employ (4.39), replace, how-

ever, $d_1(t)$ by $d_1(t+1)$ to make use of $s(t+1)$ knowledge. Figure 72 illustrates a typical dynamic hedging result. Notice that the final hedging error $e(T)$ is positive for SMPC whereas negative for Δ-hedging. This behavior could be observed frequently, see Figure 70 and Table 11 for the hedging results for all 105 stock prices considered. By the definition of (4.39), Δ-hedging always constrains $u(t)$ to lie between 0 and 1. For the SMPC formulation this is not the case. Shortselling and unconstrained ($u(t) > 1$) buying commands spontaneously result from solving the SMPC optimization problems, see Figure 72 around $t = 15$ for an illustration. Note that for a practical implementation of the SMPC algorithm it is recommended to add constraints such as $u_{\min}(t) \leq u(t) \leq u_{\max}(t)$, whereby the bounds have to be determined according to the requirements of the party writing the option. In the simplest case, $0 \leq u(t) \leq 1$, $\forall t$. As Table 11 indicates, when analyzing all 105 components of the NASDAQ-100 composite and assuming perfect knowledge of the one-step ahead underlying price $s(t+1)$, a significant wealth shortfall of $\min(e(T)) = -45.7$ resulted for Δ-hedging, whereas $\min(e(T)) = -0.6$ for LP-CVaR.

To summarize, when comparing the three stock-price scenario generation methods (logn, SVR and pert) we found the pert scheme to perform best within our SMPC-setting. Moreover, the combination of perfect $s(t+1)$ information and a SMPC algorithm results in consistently excellent final hedging errors and significantly outperforms common Δ-hedging. This finding encourages the employment of the presented SMPC algorithms and emphasizes the importance of accurately predict $s(t+1)$ at time t, as one would expect.

Barrier Option

For the UP-AND-OUT option we assumed a low barrier of $s_u = 1.1s(0)$. This resulted in 62.9% of all stock trajectories that the barrier was reached for at least one time instant before expiration date T. Simulation results are summarized in Tables 12 and 13. See also Figures 73 and 74. As for the European call option, the combination of perfect $s(t+1)$ information and a SMPC algorithm outperformed Δ-hedging. Most importantly,

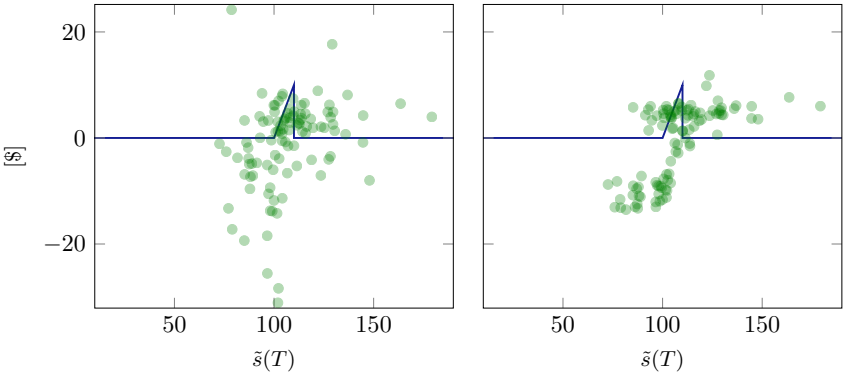


Figure 73: Results for a path-dependent Barrier option in the causal case. The normalized portfolio wealth $\tilde{w}(T) = 100w(T)/K$ and corresponding $\tilde{s}(T) = 100s(T)/K$ for all 105 stocks (green dots) is compared to the normalized payoff $\tilde{p}(T)$ (solid blue) at expiration date T. (Left) Solution for LP-CVaR, $M = 100$ and pert, see the second row from the top in Table 12. (Right) Solution for Δ-hedging, see the last row from the top in Table 12.

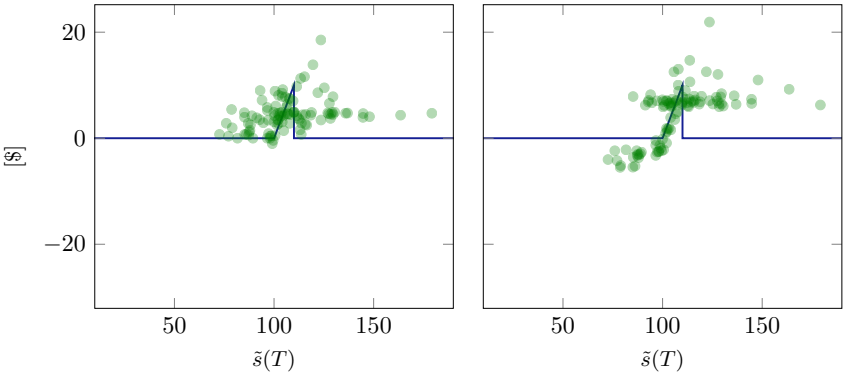


Figure 74: Results for a path-dependent Barrier option in the noncausal case. The normalized portfolio wealth $\tilde{w}(T) = 100w(T)/K$ and corresponding $\tilde{s}(T) = 100s(T)/K$ for all 105 stocks (green dots) is compared to the normalized payoff $\tilde{p}(T)$ (solid blue) at expiration date T. (Left) Solution for LP-CVaR, $M = 100$ and pert, see the second row from the top in Table 13. (Right) Solution for Δ-hedging, see the last row from the top in Table 13.

Table 12: Results for a path-dependent Barrier option in the causal case without knowledge of exact stock price $s(t+1)$ at time t. For LP-CVaR and $M = 1000$ no solution could be returned for some of the stocks considered.

Controller	M	$\mathbf{E}[e(T)]$ logn/SVR/pert	min(e(T)) logn/SVR/pert	$\mathbf{Var}[e(T)]$ logn/SVR/pert	τ, ms
QP-Var	100	$-0.34/-4.8/3.0$	$-33.2/-95.0/-27.3$	421.5/178.1/152.9	3.0
LP-CVaR	100	$-2.1/-2.5/1.9$	$-38.9/-29.0/-25.9$	119.7/81.6/148.6	3.8
LP-MinMax	100	$-2.9/-3.8/0.5$	$-45.9/-49.2/-23.0$	123.4/85.2/158.6	3.0
QP-Var	1000	$0.2/-3.1/3.3$	$-29.1/-84.1/-27.2$	447.4/153.4/172.9	12.6
LP-CVaR	1000	$-/-/-$	$-/-/-$	$-/-/-$	–
LP-MinMax	1000	$-2.1/-1.4/-0.7$	$-46.3/-25.5/-35.0$	124.5/75.2/142.2	9.8
Δ-hedging		0.7	-33.3	111.7	$6e-2$

Abbreviations: LP, linear program; LP-CVaR, linear program conditional value at risk; QP-Var, quadratic program variance; SVR, support vector regression.

Table 13: Results for a path-dependent Barrier option in the *noncausal* case with perfect knowledge of exact stock price $s(t+1)$ at time t. In accordance with the scenario generation of (4.40), we set $M = 100$.

Controller	$\mathbf{E}[e(T)]$	min(e(T))	$\mathbf{Var}[e(T)]$	τ, ms
QP-Var	6.3	-5.1	373.6	3.1
LP-CVaR	7.0	-3.1	400.3	3.8
LP-MinMax	7.0	-3.4	401.8	3.2
Δ-hedging	5.8	-7.8	192.1	$4e-2$

Abbreviations: LP, linear program; LP-CVaR, linear program conditional value at risk; QP-Var, quadratic program variance.

mainly positive final hedging errors could be recorded. As visualized by Figure 74, when analyzing all 105 stocks, the percentage of positive final hedging error was 90.5% and 62.9% for LP-CVaR and Δ-hedging, respectively. The rebalancing policy for Δ-hedging follows (4.39) and sets $u(t+\tau) = 0,\ \forall \tau \in [0, T-t]$ as soon as the underlying stock price reaches the barrier at time t. Consequently, at time t all wealth is transfered towards the risk-free asset.

Before concluding, we report two additional experiments. The first is motivated by the nature of Barrier options. By definition, as soon as the underlying stock exceeds the barrier limit the option value drops to

Table 14: Results for a European call option in the noncausal case with perfect knowledge of exact stock price $s(t+1)$ at time t. The average results for the entire NASDAQ-100 are reported. LP-MinMax is employed as SMPC algorithm. For $\sigma_{\text{pert}} < 0.06$, no solution could be returned anymore for some of the stocks considered (the solver failed to find a solution).

$(M, \sigma_{\text{pert}})$	$\mathbf{E}[e(T)]$	$\min(\mathbf{e}(\mathbf{T}))$	$\mathbf{Var}[e(T)]$	τ, ms
$(2, 0.06)$	-0.56	-339.2	1264.2	2.3
$(2, 0.3)$	-6.0	-1749.4	36547.3	2.3
$(100, 0.06)$	3.5	-0.8	121.8	3.1
$(100, 0.3)$	3.3	-0.9	123.8	3.0

Abbreviations: LP, linear program; SMPC, stochastic model predictive control.

0. A corresponding wealth and control trajectory are displayed in Figure 75 with a characteristic $w(T) > 0$. Suppose for a theoretical reason one may still want to decrease the final headging error $e(T)$ close to zero, and thereby frequently reducing excess wealth accumulated at time of barrier reaching. Then, for both the causal and noncausal setting, this can be achieved by adding perturbation noise, i.e., by setting $p^j(t+1) = 0 + 0.3\eta^j(t),\ \eta^j(t) \sim \mathcal{N}(0,1)$ for option price scenarios, where coefficient 0.3 was chosen from experiments. As Figure 75 shows, the resulting control signal $u_{\text{pert},0}(t)$ is very jagged and requires shortselling. Nevertheless, it is capable of reaching $e(T) = 0$ even in a *causal* setting. This behavior displays the powerful tracking capabilities of the SMPC algorithm, which may not only be exploited for dynamic hedging but also for index replication [29] and target performance tracking [123].

The concluding experiment is to stress the necessity of stochasticity and a sufficiently large number of generated scenarios M, even in case of perfect *one-step* ahead knowledge of stock prices. We consider (4.40) for scenario generation of stock prices and vary both M and the perturbation noise parameter σ_{pert}. Simulation results are summarized in Table 14. For visualization, see Figure 76. Note the sensitivity of control trajectories for $\sigma_{\text{pert}} = 0.01$ and the resulting temporary catastrophic tracking accuracy despite perfect one-step ahead knowledge.

Finally, we remark some success ratios reported in the literature for

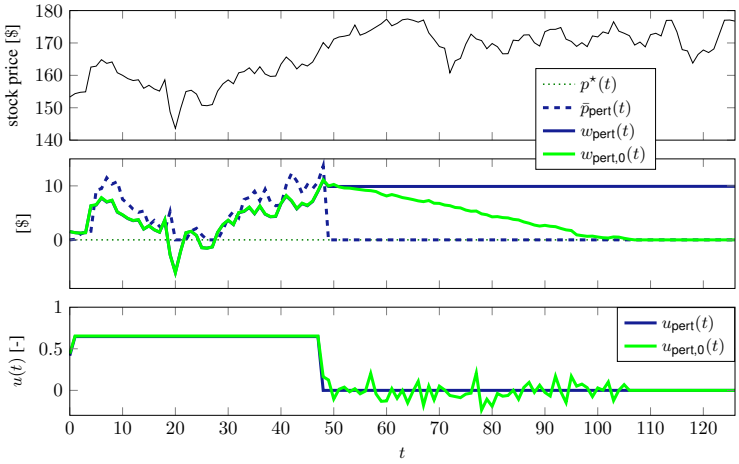


Figure 75: Employing QP-Var as the SMPC algorithm for a path-dependent Barrier option in the causal setting. See Section 4.1.6 for the reduction of the final hedging error by means of perturbation noise. The underlying stock price (top frame) is of Broadcom Ltd. (May 27 until November 25, 2016).

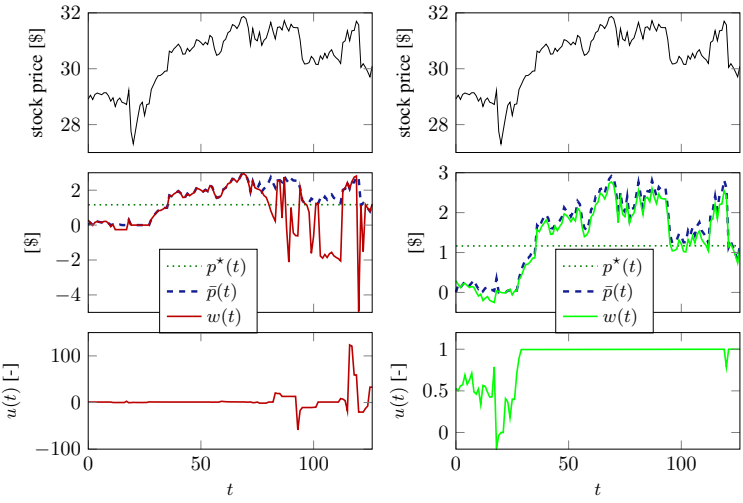


Figure 76: Noncausal case. Illustration of two different levels of perturbation in (4.40): $\sigma_{\text{pert}} = 0.01$ (left frame) and $\sigma_{\text{pert}} = 0.3$ (right frame). LP-MinMax is employed as the SMPC algorithm. The underlying stock price (top frame) is of Cisco Systems Inc. (May 27 to November 25, 2016).

correct *sign* predictions of step-ahead price difference $s(t+1)s(t)$. They are meant to underline the difficulty in generating continuously accurate step-ahead predictions in practice. In [217], support vector machines (SVM) in combination with twelve technical indicators (such as Williams %R, stochastic %K, disparity, etc.) are used to predict the direction of change in the *daily* Korea composite stock price index (KOSPI). For validation data and their best tuning parameter choices, they report a prediction performance between 50.1% and 57.8%. The same author mentioned similar results in earlier work [218].

4.1.7 Conclusions

SMPC is a suitable trading strategy for dynamic option hedging in the presence of transaction costs. The simple scenario generation method according to the pert scheme outperformed both the SVR-based and logn-model. For the noncausal and theoretical case of perfect one-step ahead knowledge of stock-price data, consistently excellent dynamic hedging performance could be observed for the SMPC algorithms, especially the LP-based LP-CVaR, significantly surpassing common Δ-hedging. These findings encourage future efforts on improving short-term stock price predictions. For a practical implementation, constraints on $u(t)$ need to be added, in the simplest case $u(t) \in [0, 1]$. SMPC can handle such input constraints naturally. In fact, as the evaluations on real-world stock prices of the NASDAQ-100 empirically show, accurate *one*-step ahead predictions in combination with the presented SMPC framework are sufficient to achieve quasi-perfect hedging. This is in stark contrast to Δ-hedging, which is not able to benefit in the same manner from perfect one-step ahead stock price predictions. The results importantly also imply that not more than accurate *one*-step ahead (instead of multiple-step ahead) predictions are required to achieve quasi-perfect hedging.

In this view, we reported hedging results based on real-world data from the NASDAQ-100, first, in the *realistic* and causal setting, and, second, in the the *optimal* setting with perfect one-step ahead stock price knowledge. As success ratios reported in the literature show, contin-

uously good step-ahead predictions are extremely difficult or impossible to achieve. Nevertheless, the two discussed settings let one interpolate the potential of SMPC for different step-ahead stock price prediction qualities that improve upon the discussed pert scheme.

An additional practical benefit of the SMPC-approach is its ability to easily incorporate a variety of constraints in the optimization problem formulations. Within the SMPC framework, we discussed the importance of scenario generation and perturbation noise, even in the case of perfect one-step ahead knowledge of stock prices.

4.1.8 Hierarchical Controller Parametrization

For the presented SMPC method for dynamic option hedging with transaction costs, there are three hierarchical layers. These are:

1. The generation of M scenarios of future market states by one of the three discussed methods: logn, SVR or pert;

2. A pricing engine the generate corresponding M future option prices according to one of the two discussed methods in Section 4.1.5;

3. The solution of one out of three SMPC problems:
   QP-Var, LP-CVaR or LP-MinMax.

4.2 Parallel Investments in Multiple Call and Put Options

This section summarizes [150]:

- M. Graf Plessen, and A. Bemporad, "Parallel investments in multiple call and put options for the tracking of desired profit profiles," in *IEEE American Control Conference*, pp. 1091-1096, 2017.

A hierarchical algorithm is presented for optimally and automatically combining various option investments to cost-efficiently realize a desired profit-vs.-underlying-price profile (profit profile). The algorithm assumes that a user-defined reference shape is defined and a set of plain vanilla options in which long and short investment positions can be taken are given. Within the presented framework, the desired profit profile can be of arbitrary piecewise-affine (PWA) shape. Depending on future underlying price predictions, it typically represents a bearish or bullish market outlook, or displays bi-modal shape for conditional market outlooks. The method provides a tool for portfolio optimization that is flexible enough to trade off different user-preferences such as exploiting on conditional market outlooks, realizing leverage, and most notably guaranteeing predictable worst-case losses for risk-minimization.

4.2.1 Introduction

The focus of this work is to develop a method for how to cost-efficiently take parallel investements in (potentially) multiple call and put options for the tracking and realization of desired profit profiles. See [191] for background on options, futures and other derivatives. The motivation for this work is the usage of the proposed tool in portfolio optimization. Standard Markowitz portfolio selection as in [265] trades-off the mean and variance of the return. For the influence of linear and fixed transaction costs in the Markowitz framework, see [257], where additionally *shortfall risk constraints* are discussed, preserving convexity of the portfolio optimization problem, making, however, the assumption of a jointly Gaussian distribution of asset returns, and not guaranteeing a bound on

the worst-case loss. In practice, observed returns frequently reveal "fat tails", i.e., higher probabilities for high price fluctuations. This motivates to look at portfolio optimization in terms of desired *profit profiles*, most notably guaranteeing predictable worst-case losses and profiting upon various market evolutions. Here, we do not discuss wealth dynamics that render a closed-loop control system and are characteristic for portfolio optimization. Instead, we present a method to realize an investment in a desired *profit profile* as an alternative to buying a specific stock or the purchase of a single option type. To realize such investments, at every portfolio rebalancing instant, static (i.e., independent between different sampling times) optimization problems are solved exploiting a given set of plain vanilla options.

While there exist well-known option strategies combining multiple options (such as the *bull call spread*, the *iron condor* and the like), see [347], [76], [88], [271], [272], [288], the novel contribution is to present a general *optimization-based* method for the cost-efficient and automated realization of an *aritrarily-shaped* PWA desired profit profile given a database of available option investments. Within the context of portfolio optimization, the presented tool allows one to concentrate on price predictions of the underlying asset and the design of desired profit profiles.

4.2.2 Call and Put Options

Employing Options

In financial terms, *securities* refer to tradable financial assets such as stocks, bonds and options. A *derivative* is a security where the value of the derivative depends explicitly on the value of another so-called *underlying* security. Here, the derivatives of interest are *options* whose underlying can be bought or sold, such as, e.g., a stock. Derivatives are standardizedly traded on *option exchanges*, e.g., on the Chicago Board Options Exchange (CBOE), or *over-the-counter* (OTC) for tailored contracts between investment parties.

There are two main types of vanilla options: a *call/put option* gives the holder the right to buy/sell the underlying asset at a given *expiration*

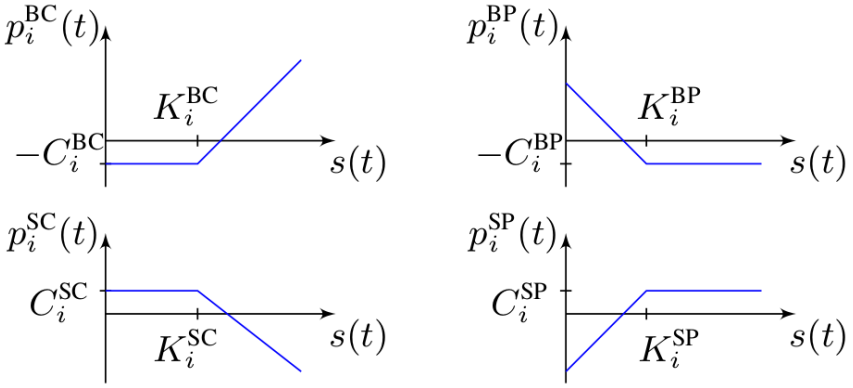
date T in the future for a predetermined *strike price*. *American*-style or simply *American* options allow to *exercise* (i.e., to buy/sell the underlying asset) at any time before and including the expiration date. In contrast, a *European*-style or simply *European* option allows the exercise right only on the expiration date. More exercise styles exist. The party who agreed to *buy* or *sell* an option is said to be *long* or *short*, respectively. We refer to *uncovered* options if the seller of an option does *not* holds a position in the underlying. The opposite are *covered* options.

Four General Option Investment Types

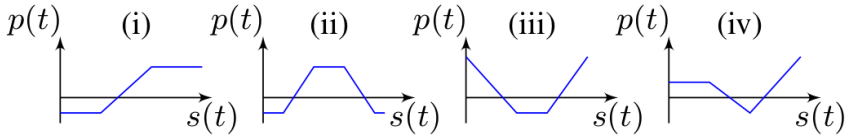
The profit equations of four general option investment types are

$$\begin{aligned} p_i^{\text{BC}}(t) &= \max(s(t) - K_i^{\text{BC}}, 0) - C_i^{\text{BC}},\ i = 1, \ldots, N^{\text{BC}}, \\ p_i^{\text{BP}}(t) &= \max(K_i^{\text{BP}} - s(t), 0) - C_i^{\text{BP}},\ i = 1, \ldots, N^{\text{BP}}, \\ p_i^{\text{SC}}(t) &= C_i^{\text{SC}} - \max(s(t) - K_i^{\text{SC}}, 0),\ i = 1, \ldots, N^{\text{SC}}, \\ p_i^{\text{SP}}(t) &= C_i^{\text{SP}} - \max(K_i^{\text{SP}} - s(t), 0),\ i = 1, \ldots, N^{\text{SP}}, \end{aligned} \tag{4.41}$$

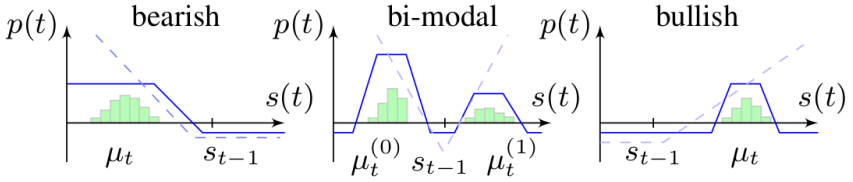
where the time index is indicated by $t \in \mathbb{Z}_+$, associated with sampling time T_s such that time instances can be described as tT_s, whereby T_s may be, for example, a trading period of one day or one month. Superscripts BC, BP, SC and SP denote "buy call option" (take a long position), "buy put", "sell call" and "sell put". For visualization of (4.41), see Figure 77. Let $y_i \in \{0, 1, 2, 3\}$ denote one of the four types. Profit from an investment is indicated by $p(t)$, e.g., $p_i^{\text{BC}}(t)$ denotes the profit at time t when holding a long position in the i-th of a set of N^{BC} call options for a specific underlying of price $s(t)$. Strike prices and costs of the option are defined by $K_i > 0$ and $C_i > 0$, respectively. We interchangeably use $s(t)$ and s_t when explicitly referring to time indices. Holding multiple option positions simultaneously then results in overall profit $p(t) = \sum_{i=1}^{N^{\text{BC}}} n_i^{\text{BC}} p_i^{\text{BC}} + \sum_{i=1}^{N^{\text{BP}}} n_i^{\text{BP}} p_i^{\text{BP}} + \sum_{i=1}^{N^{\text{SC}}} n_i^{\text{SC}} p_i^{\text{SC}} + \sum_{i=1}^{N^{\text{SP}}} n_i^{\text{SP}} p_i^{\text{SP}}$, whereby $n_i^{\text{BC}}, n_i^{\text{BP}}, n_i^{\text{SC}}, n_i^{\text{SP}} \in \mathbb{Z}_+$ denote the integer-valued number of options held and we omitted time-indices t for brevity.



(a) Illustration of four option investment types, see (4.41).



(b) Examples of option combinations: (i) *bull call spread*, (ii) *iron condor*, (iii) *long strangle*, (iv) *call backspread*.



(c) Three market outlook scenarios: bearish, bi-modal and bullish. For each scenario, two exemplary profit profiles are displayed (blue solid and dashed lines). The (scaled) PMFs of expected underlying prices, which may be used as guideline to design specific profit profiles, are displayed as green bars.

Figure 77: Illustration of various profit profiles.

Underlying Price Probability Mass Function

At time $t-1$, predictions about the future underlying price s_t can be made. We therefore describe s_t as a discrete random variable (DRV) with a (discrete) probability mass function (PMF) $f_{s_t}(s) \geq 0,\ \forall s \in \mathcal{S}_{s_t}$, which may in general be multi-modal. Here, our focus is on uni- and bi-modal distributions. The interest in uni-modal PMFs is natural due to bearish or bullish market outlooks. For the interest in bi-modal PMFs, consider the situation in which an investor is expecting a strong movement of the underlying price dependent on an earning report to be announced soon, but is uncertain about the movement direction. Predictions of the underlying price typically serve as prerequisite for the generation of a desired reference profit profile. Note, however, that the proposed algorithm allows tracking of *arbitrary* PWA profit profiles. Thus, estimated PMFs are not limiting trackable profit profiles, but merely can help in the design thereof. Assuming a uni-modal distribution, we may just consider the expected underlying price, here abbreviated by $\mu_t = \sum_{s \in \mathcal{S}_{s_t}} s f_{s_t}(s)$. For the bi-modal case, as an alternative to the complete PMF $f_{s_t}(s)$, we may just consider the conditional PMF denoted by $f_{s_t|z_t}(s|z) \geq 0$ with binary variable $z \in \{0,1\}$ indicating one of two possible event outcomes, i.e., causing a decline ($z=0$) or a rise ($z=1$) in the underlying price. Naturally, it holds $f_{s_t}(s) = \sum_{z \in \{0,1\}} f_{s_t|z_t}(s|z) f_{z_t}(z)$. We abbreviate $\mu_t^{(z)} = \sum_{s \in \mathcal{S}_{s_t}} s f_{s_t|z_t}(s|z),\ \forall z \in \{0,1\}$.

4.2.3 High-level Algorithm

There exist option strategies, see e.g. [347], that combine (superimpose) multiple options to generate *profit profiles*, see Figure 77 for illustration. Depending on the market outlook, specific selections are preferable. Our proposed high-level algorithm for profit profile generation and realization is summarized in Algorithm 5. Let us discuss the first three substeps. Step 4 is treated in all of Section 4.2.4.

Algorithm 17: Profit profile realization @$t-1$

1 **Input**: underlying price s_{t-1}, and database $\mathcal{D}$ @$t-1$.

2 **Predict future underlying price**: given past financial time-series until $t-1$, predict at least μ_t in case of a bearish, bullish or neutral market outlook, or $\mu_t^{(0)}$ and $\mu_t^{(1)}$ for a conditional market outlook; ideally, predict arbitrarily accurate the corresponding underlying price PMFs, see Section 4.2.2.

3 **Generate desired profit profile**: design a desired PWA $p^{\text{ref}}(s)$ according to (4.43) by

- deciding upon a desired *shape*, e.g., according to Table 15.
- constructing $p^{\text{ref}}(s)$ considering slope, plateau levels, and kink points selections.

4 **Solve optimization problem**: solve (4.51) for $n^\star \in \mathbb{Z}_+^{N_n \times 1}$ according to Section 4.2.4.

5 **Wait until next rebalancing time**: initiate/terminate option investment positions according to $n^\star$.

Step 1

A database of available option investment positions can be summarized as

$$\mathcal{D} = \begin{bmatrix} \{K_i^{\text{BC}}\}_{i=1}^{N^{\text{BC}}} & \{C_i^{\text{BC}}\}_{i=1}^{N^{\text{BC}}} & \{y_i^{\text{BC}}\}_{i=1}^{N^{\text{BC}}} \\ \{K_i^{\text{BP}}\}_{i=1}^{N^{\text{BP}}} & \{C_i^{\text{BP}}\}_{i=1}^{N^{\text{BP}}} & \{y_i^{\text{BP}}\}_{i=1}^{N^{\text{BP}}} \\ \{K_i^{\text{SC}}\}_{i=1}^{N^{\text{SC}}} & \{C_i^{\text{SC}}\}_{i=1}^{N^{\text{SC}}} & \{y_i^{\text{SC}}\}_{i=1}^{N^{\text{SC}}} \\ \{K_i^{\text{SP}}\}_{i=1}^{N^{\text{SP}}} & \{C_i^{\text{SP}}\}_{i=1}^{N^{\text{SP}}} & \{y_i^{\text{SP}}\}_{i=1}^{N^{\text{SP}}} \end{bmatrix} \in \mathbb{R}_+^{N_n \times 3}, \tag{4.42}$$

with $N_n = (N^{\text{BC}} + N^{\text{BP}} + N^{\text{SC}} + N^{\text{SP}})$, whereby we abbreviate $\mathcal{D} = \begin{bmatrix} \mathcal{K}^{\mathcal{D}} & \mathcal{C}^{\mathcal{D}} & \mathcal{Y}^{\mathcal{D}} \end{bmatrix}$, and where the last column indicates one of the four types of options. Note that elements of database $\mathcal{D}$ must be synchronized according to expiration dates, which becomes relevant when mixing European and American exercise style options.

Step 2

The *prediction* of future underlying prices is crucial when taking any financial investment decisions. Predictions may be based on financial ac-

countancy or technical chart analysis. For a method based on support vector machines, see [217]. There exists a plethora of approaches for *financial times-series prediction*. They are here not not our focus. We concentrate on profit profile designs and their optimization-based realizations by means of option combinations.

Step 3

We denote a desired PWA reference profit profile by

$$p^{\text{ref}}(s) = a_j s + b_j,\ s \in [K_j, K_{j+1}],\ \forall K_j \in \mathcal{K}^{\text{ref}}, \tag{4.43}$$

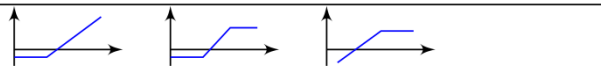
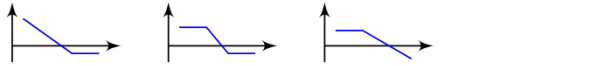
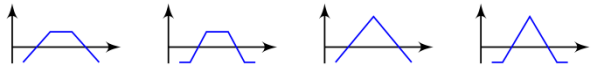
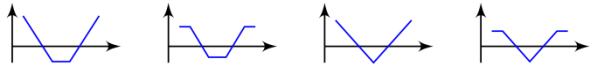
for all $j = 0, 1, \ldots, N^{\text{ref}}$, whereby the underlying price segments are defined by the set $\mathcal{K}^{\text{ref}}$. Any arbitrary PWA function is admissible. The design can be regarded as *engineering art* and is subject to user-preferences, see Figure 77. Various slope rates can be achieved by adjusting the number of options sold or bought.

Let us discuss a heuristic design of $p^{\text{ref}}(s)$. For the realization of different market outlooks and risk/reward demands, we summarize typical $p(s)$-schemes in Table 15. We remark that some profiles exhibit *plateau levels*, defined by constant p for consecutive s, and with $\min\left(p(s)\right) > -\infty$ and $\max\left(p(s)\right) < +\infty$. In case the designated $p(s)$ profile is selected to have limited downside risk, i.e., $\min\left(p(s)\right) > -\infty$, we scale $p^{\text{ref}}(s)$ such that $\min\left(p^{\text{ref}}(s)\right) = 0$ and introduce an optimization *slack variable* responsible for constant cost-efficient offset as later discussed in Section 4.2.4. For simplicity, we may assign $\frac{dp^{\text{ref}}(s)}{ds} \in \{0, 1\}$, i.e., permit only two possible slope rates. Naturally, other slope rates are possible, too.

For uni-modal market outlooks (bullish, bearish or neutral), we define $p^{\text{ref}}(s) \geq \tilde{p}_t,\ \forall s \in [\alpha\mu_t, \mu_t + (1-\alpha)\mu_t]$ with parameters $\alpha \in (0, 1]$ and $m_t \geq 0$ such that $\tilde{p}_t = m_t(\alpha\mu_t - s_{t-1})$. Figure 78 illustrates a corresponding example in which we define $m_t = 1$, $\mathcal{K}^{\text{ref}} = \{0, K_1, K_2, K_3, K_4\} = \{0, \alpha\mu_t - \tilde{p}_t, \alpha\mu_t, \mu_t + (1-\alpha)\mu_t, \mu_t + (1-\alpha)\mu_t + \tilde{p}_t\}$ and

$$p^{\text{ref}}(s) = \begin{cases} 0, & 0 \leq s \leq K_1 \text{ and } K_4 \leq s < \infty, \\ s - K_1, & K_1 \leq s \leq K_2, \\ \tilde{p}_t, & K_2 \leq s \leq K_3, \\ \tilde{p}_t - s + K_3, & K_3 \leq s \leq K_4. \end{cases}$$

Table 15: Library of typical shape designs for $p(s)$ that may be used for the realization of different market outlooks and risk/reward demands. For a discussion of corresponding kink points, slopes and plateau levels selections, see Step 3 and Figure 78.

Outlook	Sketches
bullish	
bearish	
neutral	
conditional	

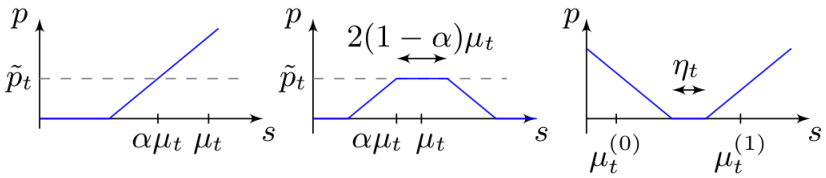


Figure 78: Illustration of heuristic methods for the PWA generation of $p^{\text{ref}}(s)$ including plateau level and kink points selections. We have $\eta_t = \alpha^{(1)}\mu_t^{(1)} - (1+\alpha^{(0)})\mu_t^{(0)}$.

For conditional market outlooks, we select tuning parameters $\alpha^{(0)}, \alpha^{(1)} \in (0, 1]$. To give an example with respect to Figure 78, we define $\mathcal{K}^{\text{ref}} = \{0, K_1, K_2\} = \{0, (1+\alpha^{(0)})\mu_t^{(0)}, \alpha^{(1)}\mu_t^{(1)}\}$ and

$$p^{\text{ref}}(s) = \begin{cases} K_1 - s, & 0 \le s \le K_1, \\ 0, & K_1 \le s \le K_2, \\ s - K_2, & K_2 \le s < \infty. \end{cases} \tag{4.44}$$

It remains to discuss the selections of upper plateau levels for conditional market outlooks and desired limited upside rewards (for overall cost reduction). Saturating (4.44), we may, for example, select two different plateau levels $\tilde{p}_t^{(0)} = \alpha^{(0)}\mu_t^{(0)}$, $\forall 0 \le s \le \mu_t^{(0)}$ and $\tilde{p}_t^{(1)} = (1-\alpha^{(1)})\mu_t^{(1)}$, $\forall s \ge \mu_t^{(1)}$, thereby trading-off different likelihoods of upside

or downside market outlooks.

4.2.4 Optimization Problem Formulation

Preparation

Vector set $\mathcal{K}^{\text{ref}}$ defines all *kink points* (s_t-coordinates at which $p^{\text{ref}}(s)$ is continuous but with discontinuous gradient) of $p^{\text{ref}}(s)$. Likewise, $\mathcal{K}^{\mathcal{D}}$ describes a similar set, see Step 1 of Section 4.2.3. In a first step, we unite and sort them according ascending s_t-coordinate, thereby creating the vector set

$$\mathcal{K} = \text{sort}\left(\mathcal{K}^{\mathcal{D}} \cup \mathcal{K}^{\text{ref}}\right), \tag{4.45}$$

where we denote the number of elements by $N_{\mathcal{K}}$. Then, all of options of (4.41) (each PWA with *one* kink point) can be cast into general PWA form with $N_{\mathcal{K}}$ kink points then common to all, i.e.,

$$p_i(s) = \begin{cases} (f_{i,0}s + g_{i,0} + c_i)n_i, & s \in [0, K_1], \\ (f_{i,1}s + g_{i,1} + c_i)n_i, & s \in [K_1, K_2], \\ \vdots & \\ (f_{i,N_{\mathcal{K}}}s + g_{i,N_{\mathcal{K}}} + c_i)n_i, & s \in [K_{N_{\mathcal{K}}}, \infty), \end{cases}$$

with $n_i \in \mathbb{Z}_+$ the number of option i for all $i = 1, 2, \ldots, N_n$. Combining all options by superposition, we obtain

$$\begin{aligned} p(s) &= \begin{cases} \sum_{i=1}^{N_n}(f_{i,0}s + g_{i,0} + c_i)n_i, & s \in [0, K_1], \\ \sum_{i=1}^{N_n}(f_{i,1}s + g_{i,1} + c_i)n_i, & s \in [K_1, K_2], \\ \vdots & \\ \sum_{i=1}^{N_n}(f_{i,N_{\mathcal{K}}}s + g_{i,N_{\mathcal{K}}} + c_i)n_i, & s \in [K_{N_{\mathcal{K}}}, \infty), \end{cases} \\ &= (sf_j^{\mathsf{T}} + g_j^{\mathsf{T}} + c^{\mathsf{T}})n, \; s \in [K_j, K_{j+1}], \; \forall K_j \in \mathcal{K}, \end{aligned} \tag{4.46}$$

for all $j = 0, 1, \ldots, N_{\mathcal{K}}$. Likewise, we cast $p^{\text{ref}}(s)$ from (4.43) into general PWA form with the same $N_{\mathcal{K}}$ kink points, then denoted by

$$p^{\text{bound}}(s) = a_j^{\text{bnd}}s + b_j^{\text{bnd}}, \; s \in [K_j, K_{j+1}], \; \forall K_j \in \mathcal{K}, \tag{4.47}$$

for all $j = 0, 1, \ldots, N_{\mathcal{K}}$, and further define

$$\tilde{b}_j = a_j^{\text{bnd}}K_j + b_j^{\text{bnd}}, \; \forall K_j \in \mathcal{K}, \; \forall j = 0, 1, \ldots, N_{\mathcal{K}}. \tag{4.48}$$

Proposition 12. *To ensure that* (4.47) *serve as a lower bound on the desired profit profile* $p(s)$ *described in* (4.46)*, it suffices to just evaluate at the kink points and constrain*

$$(K_j f_j^{\mathsf{T}} + g_j^{\mathsf{T}} + c^{\mathsf{T}})n \geq \tilde{b}_j, \tag{4.49}$$

$$f_{N_{\mathcal{K}}}^{\mathsf{T}} n \geq a_{N_{\mathcal{K}}}^{bnd}, \tag{4.50}$$

for all $K_j \in \mathcal{K}$ *and* $j = 0, 1, \ldots, N_{\mathcal{K}}$.

Proof. Let us abbreviate $z_j = (K_j f_j^{\mathsf{T}} + g_j^{\mathsf{T}} + c^{\mathsf{T}})n$ for all $K_j \in \mathcal{K}$ and $j = 0, 1, \ldots, N_{\mathcal{K}}$. Because of all $N_{\mathcal{K}} + 1$ segments being PWA, w.l.o.g. we can consider any of the segments. Then, we note that any $z = p(s)$ for $s \in [K_j, K_{j+1}]$ can be described as a linear combination $z = \gamma z_j + (1-\gamma) z_{j+1}$ for $\gamma \in [0, 1]$. Assume now $z_j \geq \tilde{b}_j$, $z_{j+1} \geq \tilde{b}_{j+1}$ and Proposition 12 as stated is wrong. The proof is then by contradiction. Similarly as above, we can write $\tilde{b} = \gamma \tilde{b}_j + (1-\gamma)\tilde{b}_{j+1}$. According to our assumption there exists a $\gamma \in [0, 1]$ such that $\gamma z_j + (1-\gamma) z_{j+1} < \gamma \tilde{b}_j + (1-\gamma)\tilde{b}_{j+1}$. This can be rewritten as $\gamma(z_j - \tilde{b}_j) + (1-\gamma)(z_{j+1} - \tilde{b}_{j+1}) < 0$, which is a contradiction since all of the left-hand side is positive. Ultimately, as a constraint on the slope, (4.50) is introduced as an alternative to account for the kink point $s \to \infty$. This concludes the proof. □

With respect to the discussion of Step 1 in Section 4.2.3, we remark for the sign of costs (when options are purchased) or premiums (when options are sold) that

$$c_i = \begin{cases} +C_i^{\mathcal{D}}, & \text{if } i \in \{1, 2, \ldots, N^{\text{BC}} + N^{\text{BP}}\}, \\ -C_i^{\mathcal{D}}, & \text{if } i \in \{N^{\text{BC}} + N^{\text{BP}} + 1, \ldots, N_n\}. \end{cases}$$

Thus, to summarize, we construct a vector set (i.e., a grid) $\mathcal{K}$ of kink points according to (4.45), before organizing $c \in \mathbb{R}^{N_n \times 1}$, $f_j \in \mathbb{R}^{N_n \times 1}$, $g_j \in \mathbb{R}^{N_n \times 1}$, $\tilde{b}_j \in \mathbb{R}$ according to (4.46) and (4.48) for all $j = 0, 1, \ldots, N_{\mathcal{K}}$, and $a_{N_{\mathcal{K}}}^{\text{bnd}} \in \mathbb{R}$.

Formulation

For the cost-efficient realization of the desired profit profile, we propose the following optimization problem:

$$\max_{n,l,\sigma} \ \lambda_0 c^\mathsf{T} n - \lambda_1 \|n\|_1 + \lambda_2 l - \lambda_3 \sigma - \lambda_4 v^\mathsf{T} n \tag{4.51a}$$

$$\text{s.t.} \quad \left(K_j f_j^\mathsf{T} + g_j^\mathsf{T} + c^\mathsf{T}\right) n \geq \tilde{b}_j - \sigma, \tag{4.51b}$$

$$f_{N_\mathcal{K}}^\mathsf{T} n \geq a_{N_\mathcal{K}}^{\text{bnd}}, \tag{4.51c}$$

$$\left(K_j f_j^\mathsf{T} + g_j^\mathsf{T} + c^\mathsf{T}\right) n \geq l, \tag{4.51d}$$

$$n \geq 0,\ l \geq l_{\text{min}},\ \sigma \geq 0, \tag{4.51e}$$

$$n \in \mathbb{Z}_+^{N_n \times 1},\ l \in \mathbb{R}_+,\ \sigma \in \mathbb{R}_+, \tag{4.51f}$$

$$\forall K_j \in \mathcal{K},\ \forall j = 0, 1, \ldots, N_\mathcal{K}, \tag{4.51g}$$

where $\lambda_0, \lambda_1, \ldots, \lambda_4 \in \mathbb{R}_+$ denote penalty weights that can easily trade-off or omit (by setting the corresponding $\lambda = 0$) different objectives. The objective function (4.51a) is composed of five components. The first component denotes the maximization of accumulated fixed costs/gains for purchasing/selling of available options. The second component is introduced to encourage sparsity in the integer-valued decision vector $n \in \mathbb{Z}_+^{N_n \times 1}$. The third component results from the introduction of *slack variable* $l \in \mathbb{R}_+$ to minimize the maximal profit profile loss (minmax problem). The fourth component penalizes another slack variable, $\sigma \in \mathbb{R}_+$, introduced for softening the constraint on the lower bound on the desired profit profile (*soft constraint*), see (4.51b). The fifth component indicates costs incurred when having to cover the selling of options (e.g., purchase and transaction costs for buying the underlying as a prerequisite for selling a covered call option). Additionally, $\sum_{i=1}^{N_\mu} \lambda_{4+i} \left(K_{\mu_i} f_{\mu_i}^\mathsf{T} + g_{\mu_i} + c_{\mu_i}\right) n$ with $K_{\mu_i} \in \{K_j : K_j \in \mathcal{K},\ K_j = \mu_i \text{ modal peaks}\}$ may be added as a sixth component to maximize profit for expected underlying prices (that possibly may be multi-modal). The first inequality (4.51b) describes the aforementioned *soft constraint* on the lower bound on the desired profit profile. It is introduced since a *reasonbale* hard lower bound on the profit profile is a priori unknown since depending on $\mathcal{D}$. The second constraint

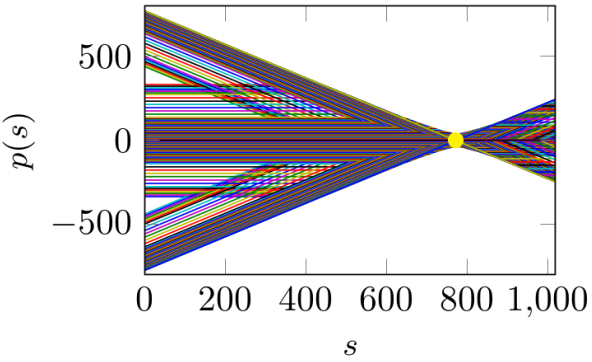


Figure 79: Illustration of $p(s)$ for all $N_n = 260$ option contracts available at the CBOE on August 24, 2016, with underlying Alphabet Inc. Class C (Nasdaq symbol GOOG), which then was quoted at 772.4$ as visualized by the yellow ball.

(4.51c) stems from (4.50). The third constraint (4.51d) results from the aforementioned introduction of slack variable l to minimize the maximal profit profile loss. In (4.51e), $l_{\text{min}} \in \mathbb{R}_+$ is defined to enforce a potential hard threshold on the maximal admissible profit profile loss. Dimensions of all optimization variables are stated in (4.51f). The coverage of all of the desired underlying price range segments is indicated by (4.51g). The solution vector of (4.51) shall be denoted by $n^\star$ and the corresponding profit profile by $p^\star(s)$.

Solution

For the solution of the mixed-integer optimization problem (4.51), we employ the domain-specific language CVXPY for optimization embedded in Python [92]. Note that when relaxing $n \in \mathbb{Z}_+^{N_n \times 1}$ to be real-valued, (4.51) is a convex problem; in fact, a linear program with an additionally added ℓ_1-norm in the objective function. All numerical experiments throughout this section were conducted on a laptop running Ubuntu 14.04 equipped with an Intel Core i7 CPU @2.80GHz×8, 15.6GB of memory, and using Python 2.7.

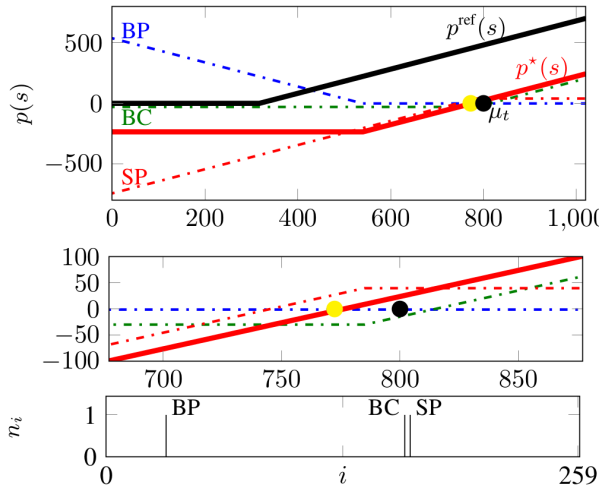


Figure 80: Results of Experiment 1 of Section 4.2.5. We selected $\mu_t = 800$ and $\alpha = 0.5$. The second subplot is a zoom-in of the first one. The third subplot indicates the number n_i and type (BC, SC, BP, SP) of options contracts sold or bought.

4.2.5 Numerical Examples

We consider real-world option price data, drawn from the CBOE at `http://www.cboe.com/delayedquote/quotetable.aspx` on August 24, 2016. As underlying, we selected Alphabet Inc. Class C (Nasdaq symbol GOOG), which was quoted with a stock price of 772.4\$ at the time of the data retrieval. The expiration date was selected to be December 16, 2016. For that datum, we retrieved the maximum amount of option data available, i.e., a total of 65 different strike prices, valued between 440\$ and 1020\$, for both call and put options. To give two examples for a call and put option with strike price 440\$, respectively: `GOOG1616L440-E` and `GOOG1616X440-E`. For C_i^{BC} and C_i^{BP}, and for C_i^{SC} and C_i^{SP}, we considered the ask and bid prices of call and put options on time of data retrieval, respectively. Thus, in total we retrieved $N_n = 260$ unique option investment opportunities as illustrated in Figure 79.

For all four experiments reported and according to (4.51), we set

$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4) = (100, 1, 1000, 100000, 0)$, $l_{\min} = -\infty$ and $m_t = 3$. The experiments differ by the selection of α, μ_t (or $\mu_t^{(0)}$ and $\mu_t^{(1)}$ for the conditional case) and desired reference profit profiles $p^{\text{ref}}(s)$. All results are visualized in Figures 80, 81, 82 and 83, and quantitatively summarized in Table 16. The fixed cost (if negative) or fixed premium (if positive) incurred at time $t-1$ is indicated by $c^\mathsf{T} n^\star$. We define percentage returns by $r^{\text{stock}}(\mu_t) = \frac{\mu_t - s_{t-1}}{s_{t-1}} 100$ for a stock investment, and, for the option investments, $r(\mu_t) = \frac{p(\mu_t)}{-c^\mathsf{T} n^\star} 100$ if $c^\mathsf{T} n^\star < 0$ (i.e., an initial expenditure was required) and $r(\mu_t) = \infty$ if $c^\mathsf{T} n^\star \geq 0$, i.e., an initial premium was received. All zero-crossings of $p^\star(s)$ are indicated by s_{BE} (*break-even* points). For the conditional (bi-modal) case, if appropriate, two quantities are stated. For simplicity, we here assumed permission to also conduct the selling of uncovered options. This assumption is justified when assuming a sufficient cash position to cover potential losses, but is to be revised when trading recursively in the context of self-financing portfolio optimization which is subject of ongoing work. With respect to database storage, we ordered options according to ascending strike prices, i.e., option identifier $i = 0, 1, 2, 3$ (see, e.g., the third subplot of Figure 80) correspond to BC,SC,BP,SP for the lowest possible strike price of $440\$$.

Several observations can be made. First, the terms in (4.51a) associated with λ_1 and λ_3 are a *must*, i.e., for ensuring sparsity in the solution vector and enabling sufficient freedom in appropriately offsetting the desired reference profit profile, respectively.

With λ_0 and λ_2 fine-tuning can be achieved in accordance with Section 4.2.4. Second, the smaller desired s-ranges for which $p^\star(s) \geq 0$ are, the smaller the worst case loss (i.e., the higher $\min(p^\star(s))$). These ranges can be controlled by selection of α. For example, selecting $\alpha = 0.98$ (instead of $\alpha = 0.95$) in the third experiment narrows the iron condor, but results in $\min p^\star(s) = -8.7$ and $\max p^\star(s) = 16.3$, see Table 16 for contrast. Thus, as accurate as possible *predictions* of underlying price evolutions are (as expected) preferable for cost minimization. Third, using CVXPY [92], remarkable differences could be observed when solving (4.51) as the original mixed-integer problem or a (real-valued) relaxed version thereof admitting $n \in \mathbb{R}_+^{N_n \times 1}$ before then rounding the solution

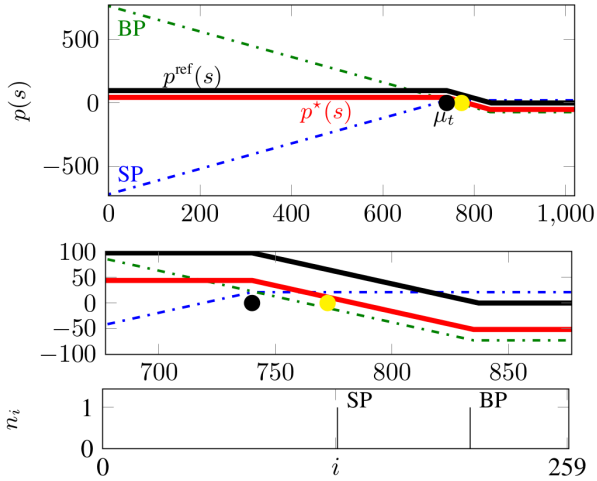


Figure 81: Results of Experiment 2 of Section 4.2.5. We selected $\mu_t = 740$ and $\alpha = 1$. Note that the desired shape is here realized optimally with two *put* options (typical for a *bear put spread*) rather than with two *call* options (*bear call spread*) which can result in the same shape, however, here at higher initial expenditure costs.

Table 16: Summary of quantitative results of the experiments from Section 4.2.5. All quantities are reported in units \$ with exception of $r(\mu_t)$ and $r^{\text{stock}}(\mu_t)$, and the computation time τ_{cvxpy} required for the solution of (4.51), which is measured in milliseconds.

	Expt. 1	Expt. 2	Expt. 3	Expt. 4
$c^T n^\star$	8.8	-51.2	11.4	-136.6
min $p(s)$	-236.2	-51.2	-13.6	-131.6
max $p(s)$	∞	43.8	11.4	∞
$p(\mu_t)$	23.8	43.8	11.4	63.4/33.4
$r(\mu_t)$	∞	85.5%	∞	46.4/24.5%
$\mu_t - s_{t-1}$	27.6	-32.4	-5.4	-102.4/87.6
$r^{\text{stock}}(\mu_t)$	3.6%	-4.2%	-0.7%	-13.3/11.3%
s_{BE}	776.2	783.8	718.6/821.4	701.7/843.3
τ_{cvxpy}	109ms	106ms	124ms	105ms

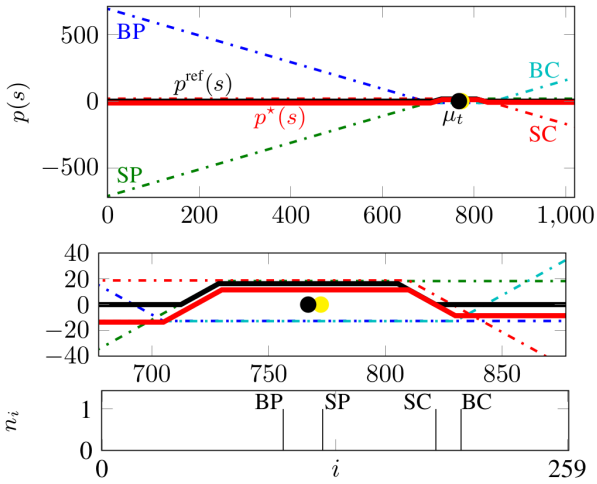


Figure 82: Results of Experiment 3 of Section 4.2.5. We selected $\mu_t = 767$ and $\alpha = 0.95$. Note that the optimal solution to (4.51) returned slightly different loss levels in case of strongly positive and strongly negative market evolutions (different plateau levels).

to the nearest integer. While the final results did only marginally (if at all) differ, computation times frequently lasted more than 12minutes vs. 100ms. All results reported in Table 16 stem from the real-valued relaxation and consequent integer-rounding solution. Ultimately, to point out a characteristic of the investment method via multiple options in parallel, consider Experiment 1. While a stock purchase requires the initial expenditure of the stock price (plus transaction costs), an investment according to Experiment 1 contrarily *generates* an initial *income*, here of $c^\mathsf{T} n^\star = 8.8\$$ per option contract, which may immediately be used to undertake other investments.

4.2.6 Conclusion

We proposed an optimization-based hierarchical algorithm for the cost-efficient and automated realization of desired profit profiles given a database of available option investments. Profit profiles can be designed arbi-

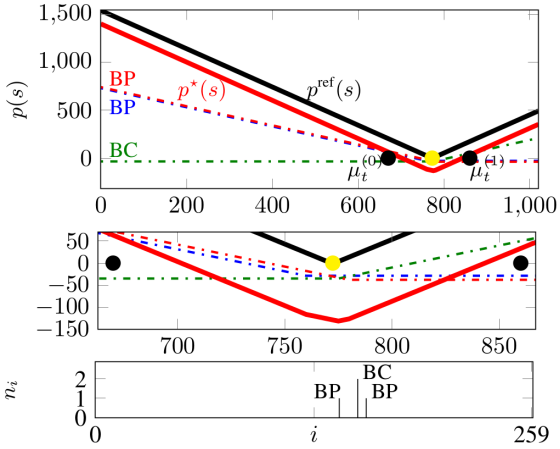


Figure 83: Results of Experiment 4 of Section 4.2.5. We selected $\mu_t^{(0)} = 670$, $\mu_t^{(1)} = 860$ and $\alpha = 0.999$. For interest, we now additionally changed the slope-rate from 1 to 2 in aim of stronger profit generation once the underlying price has passed the break-even points. The resulting profit profile is slightly asymmetrical with respect to $\mu_t^{(0)}$ and $\mu_t^{(1)}$.

trarily as piecewise-affine, typically influenced by underlying price predictions, a bullish, bearish or conditional market outlook, and accounting for user-preferences such as a bound on maximum loss. Subject of ongoing work is the incorporation of the presented framework in the context of portfolio optimization and a receding horizon control scheme.

4.2.7 Hierarchical Controller Parametrization

There are two main hierarchical layers. These are:

1. The generation of a desired PWA profit profile;
2. Its realization by means of proposed optimization problem (4.51) and a database of available option contracts.

4.3 Optimal Trading with Hindsight

This section summarizes [151]:

- M. Graf Plessen, and A. Bemporad, "A posterior multi-stage optimal trading under transaction costs and a diversification constraint," *arXiv preprint arXiv: 1709.07527*, 2017. (Submitted).

A method for the evaluation of a posteriori (historical) multi-variate multi-stage optimal trading under transaction costs and a diversification constraint is presented. Starting from a given amount of money in some currency, we analyze the stage-wise optimal allocation over a time horizon with potential investments in multiple currencies and various assets, such as, for example, assets emulating stock indices. Variants are discussed, such as unconstrained trading frequency, a fixed number of total admissible trades, and the waiting of a specific time-period after every executed trade until the next trade. Mathematical aspects are the modeling of transition dynamics as a Markov Decision Process (MDP), efficient graph generation and consequent graph search. The developed strategies are evaluated on real-world data. This work is preparatory for the labeling of data and a subsequent machine learning application.

4.3.1 Introduction

Algorithmic assistance to traders and portfolio managers is nowadays ubiquitous. We can distinguish between *algorithmic trading*, i.e., fully automated high- or low-frequency trading, and *algorithmic screening* or semi-automated high- or low-frequency trading with computer programs providing recommendations to the human trader. Both algorithmic trading and screening are fundamentally based on predictions of future developments. Predictions may be made based on, for example, financial accountancy, technical chart analysis, global macroeconomic analysis, news, sentiments and combinations thereof. There exists a plethora of literature for *financial times-series prediction*. For methods based on support vector machines, see, for example, [351], [217] and [130]. In general, influential factors on trading decisions are trading frequency, targeted

time horizons, performance expectations, asset choices, foreign exchange rates, transaction costs and risk-management, for example, in the form of investment diversification.

This work belongs to the class of technical chart analysis. The data on which the analysis is based are daily adjusted closing-prices of various currencies and assets emulating stock indices such as, e.g., the S&P 500. Short-selling, borrowing of money, and the trading of derivatives are not treated, eventhough the presented methodologies can easily be extended to include them. Starting from a given amount of money in some currency, a posteriori multi-variate multi-stage optimal trading decisions under transaction costs and a diversification constraint are reconstructed. Optimality here refers to return (wealth) maximization for a given time-period.

The following important real-world questions are addressed: what quantitative influence do different levels of transaction costs have on optimality, in particular, with respect to optimal trading frequency? How does a diversification constraint affect results? What about currency effects? Upon which increases of prices is it convenient to trade? What observations can be made, specifically, on a global scale with various currencies and markets of different volatility?

Acting in financial markets must be made by looking ahead. In contrast, this work is concerned only with *past* (real-world) data. It is meant to be preparatory work that can be used, for example, for benchmark generation in backtesting of trading strategies and labeling of data for machine learning applications. In particular, it is expected that results can be used favorably for the development and training of real-time algorithmic trading and screening systems. For example, a hierarchical closed-loop control architecture for automated trading is envisioned comprising particularly two stages: short term trajectory prediction and multi-stage investment trajectory optimization. Then, the training of the second stage is fundamentally based on one of the techniques below.

This work is related most closely[4] to [67], where dynamic programming is discussed for optimal investing in either one stock or one bond

[4]In fact, the author did not find other papers treating *a posteriori* optimal trading.

under consideration of unconstrained trading frequency, and a bound on the admissible number of trades. In addition, a method for optimization of Sterling and Sharpe ratio is presented. However, real-world price data is not analyzed. Besides real-world data, additional differences in our case are our discussion of a diversification constraint, the constraint of introducing a waiting period after every executed trade until the next trade, and a synchronous trading constraint. Furthermore, we introduce a heuristic for each of the constrained optimal investment problems (with a bound on the admissible number of trades and a waiting period constraint), thereby reducing the computational complexity of the methods while not compromising optimality of the resulting solution. For an overview of measures to reduce risk by the introduction of various constraints, for example, on the drawdown probability and shortselling, see [257] where *one-step ahead* optimization is conducted, importantly, based on estimates of one-step ahead returns and covariance matrices for a set of risky assets. In contrast, this work is concerned about *multi-stage* optimization and *historical* optimal trading with hindsight. The mathematical approaches therefore differ significantly (convex optimization vs. tree search). Optimal trading based on stochastic models and consideration of fixed and/or proportional transaction costs is treated, for example, in [7], [256], [285] and [226]. This work is data-based without consideration of any mathematical model explaining the generation of data. For a discussion about the existence of trends in financial time series, see [114].

4.3.2 One-stage Modeling of System Dynamics

System States and Time-varying Parameters

Let time index $t \in \mathbb{Z}_+$ be associated with the trading period T_s, such that trading instants are described as tT_s, whereby T_s may typically be, for example, one week, one day, one hour or less (for intraday trading). Let us define the system state z_t at time t as an eight-dimensional vector of mixed integer and real-valued quantities,

$$z_t = \begin{bmatrix} i_t & k_t & j_t & m_t^{c_t} & n_t & w_t^0 & d_t & c_t \end{bmatrix}^\mathsf{T}, \tag{4.52}$$

where $i_t \in \mathcal{I} = (\mathcal{I}_{N_c} \cup \mathcal{I}_{N_a})$ denotes investment identification numbers partitioned into N_c currencies and N_a different risky non-currency assets (such as stocks or funds emulating stock indices), such that $\mathcal{I}_{N_c} = \{0, 1, \ldots, N_c - 1\}$ and $\mathcal{I}_{N_a} = \{N_c, \ldots, N_c + N_a - 1\}$. For ease of reference, in the following, we lump currencies and non-currency assets in the term *asset* and only distinguish when context-necessary. The integer number of conducted trades along an *investment trajectory* shall be denoted by $k_t \in \mathbb{Z}$, whereby an investment trajectory is here defined as a sequence of states z_t, $t = 0, 1, \ldots, N_t$, where N_t is the time horizon length. Let j_t denote the investment identification number preceding i_t at time $t-1$ (parent node), i.e., $j_t = i_{t-1}$. We define $m_t^{c_t} \in \mathbb{R}_+$ as the real-valued and positive *cash position* (liquidity) held in the currency identified by $c_t \in \mathcal{I}_{N_c}$. The number $n_t \in \mathbb{Z}_+$ indicates the number of non-currency assets held. The current wealth, composed of cash position and non-currency asset, is denoted by w_t^0 and shall always be in monetary units EUR. We consider Euro (EUR) as the reference currency. It shall throughout this section be identified by $i_t = 0$. The integer number of time samples since the last trade is defined by $d_t \in \mathbb{Z}_+$.

Let us define a (unitless) foreign exchange (forex or fx) rate $x_t^{c_1,c_2}$ for two currencies $c_1 \in \mathcal{I}_{N_c}$ and $c_2 \in \mathcal{I}_{N_c}$ as $x_t^{c_1,c_2}$ such that

$$m_t^{c_2} = m_t^{c_1} x_t^{c_1,c_2}.$$

For example, for a foreign exchange rate between Euro and US-Dollar of EURUSD=1.1, and a conversion of 50EUR to USD, we have $c_1 = 0$ and $c_2 = 1$ (where we identified EUR and USD by currency numbers 0 and 1), $x_t^{c_1,c_2} = 1.1$ and $m_t^{c_1} = 50$ and $m_t^{c_2} = m_t^{c_1} x_t^{c_1,c_2} = 55$. We only point this out to stress $m_t^{c_1}$ and $m_t^{c_2}$ being numerical values, however, with units identified by $c_1 \in \mathcal{I}_{N_c}$ and $c_2 \in \mathcal{I}_{N_c}$.

Let us denote non-currency asset prices by $p_t^{c_t,a}$, whereby c_t identifies the price unit and $a \in \mathcal{I}_{N_a}$ the asset. We treat foreign exchange rates and asset prices as time-varying parameters obtained from data.

In the sequel, various sets of admissible system states are defined. For brevity, we therefore use a shorthand notation. For example, we define a set as $\mathcal{Z}_t = \{z_t : i_t = 10\}$, meaning $\mathcal{Z}_t = \{z_t : i_t = 10$, and i_t associated

with z_t according to (4.52)}.

Transaction Costs

For the modeling of transaction costs, we follow the notion of [257], modeling transaction costs as non-convex with a fixed charge for any nonzero trade (fixed transaction costs) and a linear term scaling with the quantity traded (proportional transaction costs). Thus, for a foreign exchange at time $t-1$, we model

$$m_t^{c_t} = m_{t-1}^{c_{t-1}} x_{t-1}^{c_{t-1},c_t}(1-\epsilon_{\text{fx}}^{c_{t-1},c_t}) - \beta_{\text{fx}}^{c_{t-1},c_t},$$

where $\epsilon_{\text{fx}}^{c_{t-1},c_t}$ and $\beta_{\text{fx}}^{c_{t-1},c_t}$ denote the linear term and the fixed charge, respectively. Examples may be $\epsilon_{\text{fx}}^{c_{t-1},c_t} = 0.01$ and $\beta_{\text{fx}}^{c_{t-1},c_t} = 50$. Similarly, transaction costs for transactions from currency to non-currency asset, between assets of different currencies and the like can be defined. We can further differentiate between linear terms for buying and selling. To fully introduce notation for transaction costs ($\epsilon_{\text{buy}}^{i_t}, \beta_{\text{buy}}^{i_t} \geq 0$), we state the transaction from a cash position towards an asset investment and vice versa. For a transaction of buying n_{t-1} of asset i_{t-1} at time $t-1$, we obtain

$$m_t^{c_t} = m_{t-1}^{c_{t-1}} x_{t-1}^{c_{t-1},c_t}(1-\epsilon_{\text{fx}}^{c_{t-1},c_t}) - \beta_{\text{fx}}^{c_{t-1},c_t} - n_{t-1}p_{t-1}^{c_t,i_t}(1+\epsilon_{\text{buy}}^{i_t}) - \beta_{\text{buy}}^{i_t}.$$

For a transaction of selling n_{t-1} of asset i_{t-1} and transforming to currency c_t, we obtain

$$m_t^{c_t} = \left(m_{t-1}^{c_{t-1}} + n_{t-1}p_{t-1}^{c_{t-1},i_{t-1}}(1-\epsilon_{\text{sell}}^{i_{t-1}}) - \beta_{\text{sell}}^{i_{t-1}}\right) x_{t-1}^{c_{t-1},c_t}(1-\epsilon_{\text{fx}}^{c_{t-1},c_t}) - \beta_{\text{fx}}^{c_{t-1},c_t}.$$

Note that transaction costs may vary dependent on the assets involved in the transformation from one to another. Further, note that we here neglect unavoidable costs such as yearly fees to financial institutes for depot keeping etc.

4.3.3 Transition Dynamics

Given our assumption of being able to invest in currencies and non-currency assets, there are six general types of transitions dependent on

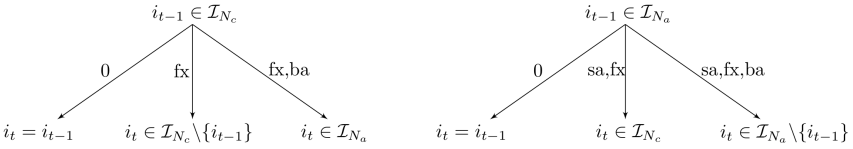


Figure 84: Abstract visualization of the transition dynamics modeled as a Markov Decision Process (MDP). Abbreviations are "0" for no action, "fx" for foreign exchange, "ba" for buying asset and "sa" for selling asset.

the investment at time $t-1$, visualized abstractly in Figure 84. For an introduction to Markov Decision Processes (MDP), see [315]. We initialize

$$z_0 = \begin{bmatrix} 0 & 0 & 0 & m_0^0 & 0 & m_0^0 & 0 & 0 \end{bmatrix}^\mathsf{T}, \tag{4.53}$$

where, for example, we set $m_0^0 = 100000$. Then, we define our transition dynamics as

$$z_t = \begin{cases} z_t^{(1)}, \text{ if } \{i_t : i_t = i_{t-1},\ z_{t-1} \text{ with } i_{t-1} \in \mathcal{I}_{N_c}\}, \\ z_t^{(2)}, \text{ if } \{i_t : i_t \in \mathcal{I}_{N_c} \backslash \{i_{t-1}\},\ z_{t-1} \text{ with } i_{t-1} \in \mathcal{I}_{N_c}\}, \\ z_t^{(3)}, \text{ if } \{i_t : i_t \in \mathcal{I}_{N_a},\ z_{t-1} \text{ with } i_{t-1} \in \mathcal{I}_{N_c}\}, \\ z_t^{(4)}, \text{ if } \{i_t : i_t = i_{t-1},\ z_{t-1} \text{ with } i_{t-1} \in \mathcal{I}_{N_a}\}, \\ z_t^{(5)}, \text{ if } \{i_t : i_t \in \mathcal{I}_{N_c},\ z_{t-1} \text{ with } i_{t-1} \in \mathcal{I}_{N_a}\}, \\ z_t^{(6)}, \text{ if } \{i_t : i_t \in \mathcal{I}_{N_a} \backslash \{i_{t-1}\},\ z_{t-1} \text{ with } i_{t-1} \in \mathcal{I}_{N_a}\}, \end{cases} \tag{4.54}$$

where $z_t^{(j)},\ \forall j = 1, \ldots, 6$, are defined next and our control variable u_{t-1} is the targeted investment identified by variable i_t, i.e., $u_{t-1} = i_t$. We have

$$\left[\ z_t^{(1)} \ \middle|\ z_t^{(2)} \ \middle|\ z_t^{(3)}\ \right] = \left[\begin{array}{c|c|c} i_{t-1} & i_t & i_t \\ k_{t-1} & k_{t-1}+1 & k_{t-1}+1 \\ j_{t-1} & j_{t-1} & j_{t-1} \\ m_{t-1}^{c_{t-1}} & \varphi(m_t^{c_t}) & \tilde{m}_t^{c_t} \\ 0 & 0 & \tilde{n}_t \\ w_{t-1}^0 & m_t^{c_t} x_t^{c_t,0} & \tilde{w}_t^0 \\ \tilde{d}_t & \tilde{d}_t & \tilde{d}_t \\ c_{t-1} & i_t & c(i_t) \end{array}\right]. \tag{4.55}$$

with $c(i_t)$ denoting the currency of asset i_t and with

$$\tilde{d}_t = \begin{cases} d_{t-1}+1, & \text{if } d_{t-1} < D-1, \\ 0 & \text{otherwise,} \end{cases}$$
$$\varphi(m_t^{c_t}) = m_{t-1}^{c_{t-1}} x_{t-1}^{c_{t-1},c_t}(1-\epsilon_{\text{fx}}^{c_{t-1},c_t}) - \beta_{\text{fx}}^{c_{t-1},c_t},$$

and where variable D determines an overflow in d_t and will become relevant when later discussing the constraint of waiting a specific amount of time until the next admissible trade. Furthermore, $\tilde{m}_t^{c_t}$ and $\tilde{n}_t$ are obtained from solving

$$\max_{m_t^{c_t}\geq 0} \Bigg\{ n_t : n_t = \frac{m_{t-1}^{c_{t-1}} x_{t-1}^{c_{t-1},c_t}(1-\epsilon_{\text{fx}}^{c_{t-1},c_t}) - \beta_{\text{fx}}^{c_{t-1},c_t} - \beta_{\text{buy}}^{i_t} - m_t^{c_t}}{p_{t-1}^{c_t,i_t}(1+\epsilon_{\text{buy}}^{i_t})}, \\ n_t \in \mathbb{Z}_+ \Bigg\}, \tag{4.56}$$

with $\tilde{m}_t^{c_t}$ denoting the optimizer and $\tilde{n}_t$ the corresponding optimal objective function value. Thus, given $m_{t-1}^{c_{t-1}}$, we find the largest possible positive integer number of assets we can purchase under the consideration of transaction costs. The (small) cash residual is then $\tilde{m}_t^{c_t} \geq 0$. Therefore, for holding value at time t in currency EUR, we obtain

$$\tilde{w}_t^0 = (\tilde{m}_t^{c_t} + \tilde{n}_t p_t^{c_t,i_t}) x_t^{c_t,0}. \tag{4.57}$$

We have

$$\left[\begin{array}{c|c|c} z_t^{(4)} & z_t^{(5)} & z_t^{(6)} \end{array}\right] = \left[\begin{array}{c|c|c} i_{t-1} & i_t & i_t \\ k_{t-1} & k_{t-1}+1 & k_{t-1}+1 \\ j_{t-1} & j_{t-1} & j_{t-1} \\ m_{t-1}^{c_{t-1}} & \phi(m_t^{c_t}) & \bar{m}_t^{c_t} \\ n_{t-1} & 0 & \bar{n}_t \\ w_{t-1}^0 & m_t^{c_t} x_t^{c_t,0} & \bar{w}_t^0 \\ \tilde{d}_t & \tilde{d}_t & \tilde{d}_t \\ c_{t-1} & i_t & c(i_t) \end{array}\right], \tag{4.58}$$

with

$$\phi(m_t^{c_t}) = \left(m_{t-1}^{c_{t-1}} + n_{t-1} p_{t-1}^{c_{t-1},i_{t-1}}(1-\epsilon_{\text{sell}}^{i_{t-1}}) - \beta_{\text{sell}}^{i_{t-1}}\right) x_{t-1}^{c_{t-1},c_t}(1-\epsilon_{\text{fx}}^{c_{t-1},c_t}) \\ -\beta_{\text{fx}}^{c_{t-1},c_t},$$

and where $\bar{m}_t^{c_t}$ and $\bar{n}_t$ are obtained from solving

$$\max_{m_t^{c_t} \geq 0} \left\{ n_t : n_t = \frac{\phi(m_t^{c_t}) - \beta_{\text{buy}}^{i_t} - m_t^{c_t}}{p_{t-1}^{c_t,i_t}(1+\epsilon_{\text{buy}}^{i_t})},\ n_t \in \mathbb{Z}_+ \right\}, \tag{4.59}$$

with $\bar{m}_t^{c_t}$ denoting the optimizer and $\bar{n}_t$ the corresponding optimal objective function value. For current holding value in EUR, we then obtain $\bar{w}_t^0 = (\bar{m}_t^{c_t} + \bar{n}_t p_t^{c_t,i_t}) x_t^{c_t,0}$.

The solution to (4.56) and (4.59) can be easily computed by setting $m_t^{c_t}$ initially zero, then rounding the corresponding real-valued n_t to the largest smaller integer, before then computing the cash residuals, respectively. The methodology of preserving the cash residual and enforcing an integer valued number of shares in assets is done with the purpose of realistic real-world modeling.

Remarks about Optimality and System Modeling

An investment trajectory was defined as a sequence of states z_t, $t = 0, 1, \ldots, N_t$. We wish to find an optimal (in the sense of wealth-maximizing) investment trajectory. Several observations with respect to above problem formulation and system modeling can be made.

First, suppose *all* of the initial money m_0^0 is fully allocated to the optimal investment trajectory, then there is no diversification constraint present. Then, the optimal investment trajectory never returns (with *return* here defined as $r_{N_t} = (w_{N_t}^0 - m_0^0)/m_0^0$) less than 0%. This is since one feasible investment trajectory is to remain invested in the initial reference currency (EUR) for all $t = 0, 1, \ldots, N_t$. This may be taken into account as a heuristic when creating the transition graph.

Second, the above system dynamics modeling naturally results in cash residuals whenever investing in a non-currency asset. According to our modeling, the cash residuals are enforced to be in the currency of the purchased asset. This may be suboptimal in very specific cases. We therefore want to further motivate our methodology in the following. Suppose the non-currency asset in which we invest is extremely expensive. Then, it may be worthwhile to invest the cash residual into another asset more profitable than the "enforced" residual currency. In the extreme

case, consider Berkshire Hethaway Inc. where *one* share costs approximately 196000 EUR. In such a case, the cash residual may be huge. Nevertheless, assets with such prices are extremely rare. Consider, for example, publicly traded shares of the Danish A.P. Moller–Maersk Group, in the past frequently element of lists summarizing the *ten* most expensive stocks *in the world* with a price of (currently) approximately 1100 EUR. Consider further that the average price of a share in a company listed in the DAX (German Stock Market Index) is currently 70 EUR, whereby the most expensive share of the 30 stocks listed in the DAX costs approximately 180 EUR. For the vast majority of shares it is therefore not worth to invest the cash residual differently than in the asset currency. All potential gains are in the vast majority of cases used up by transaction costs, in particular, by fixed transaction costs. Consider for example the case of a minimal fixed transaction cost of 50 EUR in relation to the aforementioned average price of 70 EUR. This is one reason for the formulation of transition dynamics as above. Another one is that in order to account for freely investing of cash residuals, an extension of the state space (beyond 8 variables) is required. *Whenever* buying any non-currency asset, a residual results which consequently can be invested in any of the $N_c + N_a - 1$ assets. Thus, $N_c + N_a - 1$ additional branches are added in the transition graph, which, in the most general case, can then be further branched at subsequent stages, which considerably complicates the tracking of states.

Third, transition dynamics (4.54) indicate an *all-or-nothing* strategy, i.e., at every $t \geq 0$, the investment at that time is maintained or, alternatively, reallocated to exactly *one*–the most profitable–currency or asset, whereby cash residuals are accounted for as described in the previous paragraph.

Fourth, let us briefly discuss the effect of absence of transaction costs on optimal trading frequency. For simplicity let us consider the case of being able to invest in an asset of variable value (such as a stock) and holding of a cash position in the currency in which the risky asset is traded. Relevant discrete-time dynamics can then be written as

$$w_t = m_t + n_t p_t, \quad \text{and} \quad m_t = m_{t-1} - n_t p_{t-1}, \tag{4.60}$$

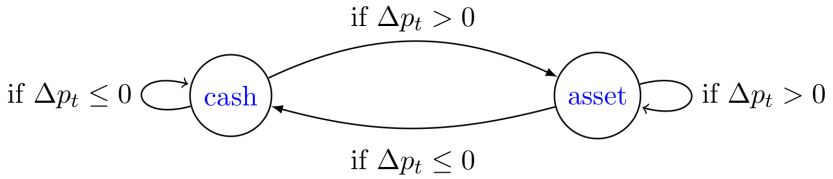


Figure 85: Visualization of the Markov decision process when trading only a cash and an asset position in the *absence* of transaction costs. The optimal trading strategy is to trade upon *any* change of Δp_t-sign, i.e., even if it is minimally small ($\Delta p_t \to 0$).

with m_t the cash position, n_t the number of shares in the risky asset, p_t the price of the asset and w_t the wealth at time t. At every time t a decision about a reallocation of investments is made. For a final time period $t = 0, 1, \ldots, N_t$, we wish to maximize $w_{N_t} - w_0$ which can be expanded as

$$w_{N_t} - w_0 = \sum_{t=1}^{N_t} w_t - w_{t-1}. \tag{4.61}$$

To maximize (4.61), we thus have to maximize the increments. Combining (4.60), we write

$$w_t - w_{t-1} = n_t(p_t - p_{t-1}) - n_{t-1}p_{t-1},$$

which therefore motivates the following optimal trading strategy, implemented at every $t-1$: if $p_t > p_{t-1}$, maximize n_t and set $n_{t-1} = 0$ (allocate maximal ressources towards the asset), and if $p_t \leq p_{t-1}$, set $n_t = 0$ and minimize n_{t-1} (i.e., sell the asset if held at $t-1$ and allocate maximal ressources towards the cash position). We profit on a price increase of the asset, and maintain our wealth on a price decrease[5]. Thus, it is optimal to trade upon *every* change of sign of $\Delta p_t = p_t - p_{t-1}$, see also Figure 85. If $\Delta p_t > 0$ at time $t-1$, we buy the asset (if we do not already hold it), and if $\Delta p_t \leq 0$, we sell the asset (if we hold it). An immediate and important observation is thus the following remark.

[5]We here assume a *long-only* strategy. By the use of derivative contracts, we could increase wealth on an underlying asset price decrease as well.

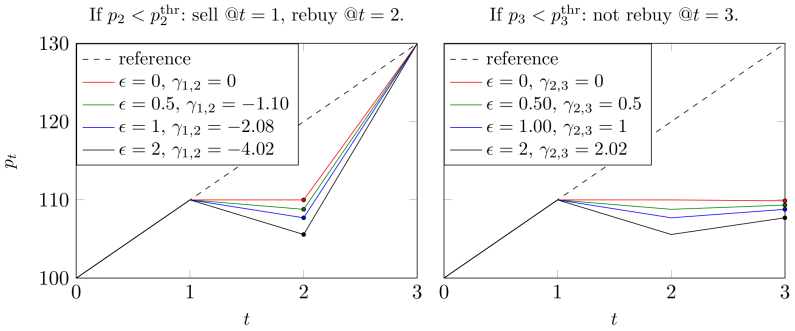


Figure 86: Influence of transaction costs on the strategy of selling and rebuying an asset instead of holding it only. Currency affects are already incorporated in asset price p_t. Transaction cost levels are indicated by ϵ.

Remark 9. *Under the absence of transaction costs, trading upon any change in sign of $\Delta p_t = p_t - p_{t-1}$ is the optimal trading policy.* □

Thus, by Remark 9, for absent transaction costs, *high-frequency trading* is always the optimal trading policy. Furthermore, $\Delta w_t = w_t - w_{t-1} \geq 0$, $\forall t = 1, 2, \ldots, N_t$. Thus, in absence of transaction costs, at *every* time step there is at least no incremental decrease in wealth when employing the optimal trading policy. Naturally, when including non-zero transaction costs, this is in general not the case anymore, i.e., we may (at least temporarily) have $\Delta w_t < 0$. In addition, optimal trading frequency will be non-trivially affected. Quantitative examples for optimal trading frequencies under transaction costs are given in Section 4.3.6.

Fifth, motivated by the previous paragraph and under the consideration of transaction costs, a valid question to address is when to sell and rebuy a non-currency asset given a long-term trend but *temporary dip* in price. Selling and rebuying may optimize profit. The situation is illustrated in Figure 86 where we consider for simplicity four (in general non-uniformly spaced) trading times. Suppose at time $t = 0$ a (non-currency) asset is bought. Given the "reference" price evolution (here assumed affine), the optimal investment trajectory is to hold the asset until $t = 3$. In contrast, for a temporary decrease in p_t the optimal profit maximizing investment trajectory is to sell the asset at $t = 1$ and rebuy it at $t = 2$ as

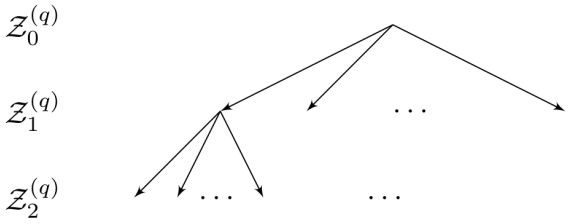


Figure 87: Abstract visualization of a multi-stage transition graph with $t = 0, 1, 2$. Superscript "$^{(q)}$" implies any of Q different transition graphs with $q = 0, 1, \ldots, Q-1$.

soon as p_2 drops below a threshold, here denoted by p_2^{thr}. The necessary decrease can be quantified by variable $\gamma_{t,t+1} = 100\frac{p_{t+1}-p_t}{p_t}$ measured in percent. We assume propotional costs (indicated in $\%$) identical for buying and selling, i.e., $\epsilon = \epsilon_{\text{buy}} = \epsilon_{\text{sell}}$. For $\epsilon = 0$, we also set $\beta = 0$. For all other cases, we set $\beta = 50$. For the transition dynamics (4.54), threshold values p_2^{thr} can be computed by iteration. Results are illustrated in Figure 86. As expected, the typical minimal decrease in p_2 w.r.t. p_1 required for the strategy of selling and rebuying being optimal is approximately twice the proportional transaction cost level. Twice because of selling *and* rebuying. Approximately because cash residuals due to integer-valued number of assets and fixed transaction costs are accounted for. Similarly, p_3^{thr} can be computed as the threshold price such that for $p_3 < p_3^{\text{thr}}$ asset rebuying is not optimal anymore.

4.3.4 Multi-stage System Dynamics Optimization Without Diversification Constraint

Multi-Stage Optimization

Multi-stage system dynamics can be modeled in form of a transition graph. We therefore assign a set $\mathcal{Z}_t$ of admissible states to every time stage t. For investment trajectory optimization without a diversification constraint, we employ *one* transition graph. For investment trajectory optimization with a diversification constraint, *multiple* transition graphs, $q = 0, 1, \ldots, Q-1$, are introduced in Section 4.3.5. Here, we therefore al-

ready introduce notation $\mathcal{Z}_t^{(q)}$, see Figure 87. However, for the remainder of this section, we drop the superscript "$^{(q)}$" and focus on optimization without a diversification constraint, i.e., $q = 0$. We define the initial set

$$\mathcal{Z}_0 = \left\{ z_0 : z_0 = \begin{bmatrix} 0 & 0 & 0 & m_0^0 & 0 & m_0^0 & 0 & 0 \end{bmatrix}^\mathsf{T} \right\}. \tag{4.62}$$

Next, three constraints on transition graph generation are discussed.

Case 1: Unconstrained Trading Frequency

Remark 10. *Suppose that following a particular investment trajectory, at time τ an investment state z_τ is reached, specifically, with a particular i_τ, w_τ^0 and $j_\tau = i_{\tau-1}$. Suppose further there exists another investment trajectory resulting in the same asset, i.e., $\tilde{i}_\tau = i_\tau$, but in contrast with $\tilde{w}_\tau^0 > w_\tau^0$ and $\tilde{j}_\tau \neq j_\tau$. Then, the former investment trajectory can be dismissed from being a possible candidate segment for the optimal investment trajectory. This is because any trajectory continuing the latter investment trajectory will always outperform a continuation of the former investment trajectory for all $t > \tau$.* □

Remark 10 motivates a simple but efficient graph generation method:

1. Branch from every state $z_{t-1} \in \mathcal{Z}_{t-1}$ to all possible states z_t at time t according transition dynamics (4.54), whereby we summarize the set of states at time $t-1$ from which z_t can be reached as $\mathcal{J}_{t-1}^{z_t}$;

2. Select the optimal transitions and thus determine $\mathcal{Z}_t$ according to

$$\mathcal{Z}_t = \left\{ z_t : \max_{j_t \in \mathcal{J}_{t-1}^{z_t}} \{w_t^0\},\ \forall i_t \in \mathcal{I} \right\}, \tag{4.63}$$

   recalling the definition $j_t = i_{t-1}$, and thereby selecting the solutions with highest value $w_t^0, \forall i_t \in \mathcal{I} = \{0, 1, \ldots, N_c + N_a - 1\}$.

The resulting transition graph holds a total of $N_z(t) = 1 + (N_c + N_a)t$ states up to time $t \geq 0$. For a time horizon N_t, the optimal investment strategy (denoted by superscript "*") can then be reconstructed by proceeding backwards as

$$z_{N_t}^\star = \left\{ z_{N_t} : i_{N_t}^\star = \max_{i_{N_t}} \{w_{N_t}^0\},\ i_{N_t} \in \mathcal{I} \right\}, \tag{4.64a}$$

$$z_{t-1}^\star = \left\{ z_{t-1} : i_{t-1} = j_t^\star \right\},\ \forall t = N_t, N_t - 1, \ldots, 1. \tag{4.64b}$$

The resulting investment trajectory is optimal since by construction of the transition graph as outlined, starting from z_0, there exists exactly one wealth maximizing trajectory to every investment $i_t = 0, 1, \ldots, N_c + N_a - 1$ for every time $t = 0, 1, \ldots, N_t$. By iterating backwards the optimal investment decisions at every time stage are determined.

Case 2: Bound on the Admissible Number of Trades

We constrain the investment trajectory to include at most $K \in \mathbb{Z}_+$ trades for time horizon $t = 0, 1, \ldots, N_t$, whereby we define a *trade* as a reallocation of an investment incurring any kind of transaction costs according to Figure 84 or any switching in the asset identification number i_t. A transition according to $i_t = i_{t-1}$ is consequently not a trade. The set of admissible states is thus generated as

$$\mathcal{Z}_t = \left\{ z_t : \max_{j_t \in \mathcal{J}_{t-1}^{z_t}} \{w_t^0\},\ \forall k_t < K \text{ and unique, and } \forall i_t \in \mathcal{I} \right\}. \quad (4.65)$$

As a result, the resulting transition graph holds a total of $N_z(t) = 1 + \sum_{l=1}^{T} (N_c + N_a) \min(l, K)$ states. The reconstruction of optimal investment decisions is similar to (4.64).

Note that the total number of states, $N_z(t)$, quickly reaches large numbers. We therefore introduce a heuristic to reduce $N_z(t)$ while not compromising optimality of the solution.

Proposition 13. *While not compromising the finding of an optimal investment trajectory, the set of admissible states $\mathcal{Z}_t$ of* (4.65) *can be shrunken to $\tilde{\mathcal{Z}}_t$ according to the following Algorithm:*

Algorithm 18: Bound on the admissible number of trades

1 Initialize: $\tilde{\mathcal{Z}}_t = \mathcal{Z}_t$.
2 **for** *every $i_t \in \mathcal{I}$ such that the corresponding $z_t \in \mathcal{Z}_t$ of* (4.65) **do**
3 Compute: $k_t^{\text{opt}}(i_t) = \left\{ k_t : w_t^{0,\text{opt}}(i_t) = \max\{w_t^0\} \text{ s.t. corresponding } z_t \in \mathcal{Z}_t \text{ of (4.65)} \right\}$.
4 Shrink: $\tilde{\mathcal{Z}}_t \leftarrow \tilde{\mathcal{Z}}_t \setminus \left\{ z_t : k_t > k_t^{\text{opt}}(i_t) \right\}$.

Proof. W.l.o.g., suppose that for a given $i_t = i \in \mathcal{I}$ we have determined $k_t^{\text{opt}}(i_t)$. Let the associated state vector be denoted by $z_t^{\text{opt}}(i_t)$. Then, we can discard all z_t with $i_t = i$ and $k_t > k_t^{\text{opt}}(i_t)$, since $w_{t+\tau}^{0,\text{opt}}(i_t) \geq w_{t+\tau}^0$, $\forall \tau \geq 0$, and the admissible set for state $z_t^{\text{opt}}(i_t)$ is thus larger by at least the option of one additional trade, in comparison to the admissible set corresponding to all $z_t \in \mathcal{Z}_t$ of (4.65) with $i_t = i$ and $k_t > k_t^{\text{opt}}(i_t)$. This concludes the proof. □

Note that the total number of states, $N_z(N_t)$, cannot be predicted anymore precisely as before for Case 1. Instead, $N_z(N_t)$ is now data-dependent. Quantitative experiments are reported in Section 4.3.6.

Case 3: Waiting Period after every Trade until the Next Trade

We constrain the investment trajectory to waiting of at least a specific time period D after every executed trade until the next trade. Thus,

$$\mathcal{Z}_t = \left\{ z_t : \max_{j_t \in \mathcal{J}_{t-1}^{z_t}} \{w_t^0\},\ \forall d_t < D \text{ and unique, and } \forall i_t =\in \mathcal{I} \right\}. \quad (4.66)$$

As a result, the resulting transition graph holds a total of $N_z(t) = 1 + \sum_{l=1}^{T} \left((N_c + N_a - 1)\min(l, D) + 1 + \min\left(\max(0, l - D), D - 1\right)\right)$ states. The reconstruction of optimal investment decisions is similar to (4.64).

Similarly to Case 2, the total number of states, $N_z(t)$, quickly reaches large numbers. We therefore also introduce a heuristic as follows.

Proposition 14. *While not compromising the finding of an optimal investment trajectory, the set of admissible states $\mathcal{Z}_t$ of* (4.66) *can be shrunken to $\bar{\mathcal{Z}}_t$ according to the following Algorithm:*

Algorithm 19: Waiting period after every trade until next trade

1 Initialize: $\bar{\mathcal{Z}}_t = \mathcal{Z}_t$.
2 **for** *every $i_t \in \mathcal{I}$ such that the corresponding $z_t \in \mathcal{Z}_t$ of* (4.66) **do**
3 Compute: $d_t^{\text{opt}}(i_t) = \left\{ d_t : w_t^{0,\text{opt}}(i_t) := \max\{w_t^0\} \text{ s.t. corresponding } z_t \in \mathcal{Z}_t \text{ of (4.66)} \right\}$.
4 Shrink: $\bar{\mathcal{Z}}_t \leftarrow \bar{\mathcal{Z}}_t \backslash \left\{ z_t : 0 < d_t < d_t^{\text{opt}}(i_t) \right\}$.

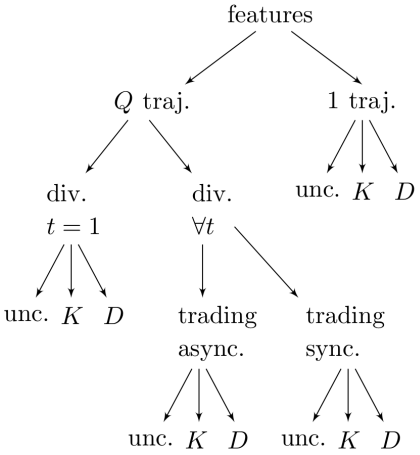


Figure 88: Abstract overview over some analysis tools. In particular, a distinction is made between inclusion of a diversification constraint ("Q traj.") and three constraints affecting trading frequency: unconstrained trading frequency ("unc."), at most K-trades ("K") along the investment trajectory for the time-horizon of interest, and the waiting of period D after execution of a trade until the next trade along every investment trajectory. Additionally, we may diversify ("div.") only at, e.g., $t = 1$. Or, we may also only trade at the same times for all investment trajectories, i.e., the trading is or is not enforced to be synchronized ("sync." or "async"). Variants are also possible, for example, implementing the diversification constraint only at a set of specific times, see Section 4.3.5.

Proof. W.l.o.g., suppose for a given $i_t = i \in \mathcal{I}$ we have determined $d_t^{\text{opt}}(i_t)$. Let the associated state vector be denoted by $z_t^{\text{opt}}(i_t)$. Then, we can discard all z_t with $i_t = i$ and $0 < d_t < d_t^{\text{opt}}(i_t)$, since $w_{t+\tau}^{0,\text{opt}}(i_t) \geq w_{t+\tau}^{0}$, $\forall \tau \geq 0$, and the admissible set for state $z_t^{\text{opt}}(i_t)$ is larger by being closer to a potential next trade by at least one trading sampling time, in comparison to the admissible set corresponding to all $z_t \in \mathcal{Z}_t$ of (4.65) with $i_t = i$ and $0 < d_t < d_t^{\text{opt}}(i_t)$. This concludes the proof. □

Similarly to Case 2, the total number of states, $N_z(N_t)$, cannot be predicted anymore precisely as before for Case 1. It is now data-dependent instead. Quantitative results are reported in Section 4.3.6. The heuristic significantly reduces computational complexity.

4.3.5 Multi-stage System Dynamics Optimization With a Diversification Constraint

In portfolio optimization, the introduction of a diversification constraint is regarded as a measure to reduce drawdown risk. There exist multiple variants of diversification constraints, see, for example, [257].

For our purpose of analysis of historical optimal trading, we first divide our initial wealth m_0 into Q parts of equal proportion. Then, we impose constraints on each of the corresponding Q investment trajectories. In the unconstrained case, all Q trajectories would coincide. Including a diversification constraint allows for different implementations, see Figure 88. It is distinguished between, first, constraints referring to relations between multiple investment trajectories (diversification at only the initial time, diversification for all times, asynchronous trading and synchronous trading), and, second, constraints referring to relations along any specific investment trajectory (unconstrained trading frequency, at most K trades admitted along the investment trajectory, and the enforcement of a waiting period after each executed trade).

We define a diversification constraint at a specific time t such that each of the states of the Q-trajectories (at that time), $z_t \in \mathcal{Z}_t^{(q)}$, $\forall q = 0, \ldots, Q-1$, must be invested *differently*. Thus, each asset identification number $i_t^{(q)}$ must be different $\forall t = 0, 1, \ldots, N_t$, $\forall q = 0, 1, \ldots, Q-1$.

We define the sets of admissible states $\mathcal{Z}_t^{(q)}$, $\forall t = 0, 1, \ldots, N_t$ and $\forall q = 0, 1, \ldots, Q-1$, sequentially and ordered according to optimality. Thus, $\mathcal{Z}_t^{(1)}$, $\forall t = 0, 1, \ldots, N_t$ is constructed accounting only for the optimal investment trajectory associated with $\mathcal{Z}_t^{(0)}$, i.e., the set $\mathcal{Z}_t^{(0),\star}$, $\forall t = 0, \ldots, N_t$, whereas $\mathcal{Z}_t^{(q)}$ is constructed accounting for all of the optimal investment trajectories associated with $\mathcal{Z}_t^{(0),\star}$, $\mathcal{Z}_t^{(1),\star}, \ldots, \mathcal{Z}_t^{(q-1),\star}$. Here, $\mathcal{Z}_t^{(q),\star}$, $\forall q = 0, 1, \ldots, Q-1$ denotes the set states at each time t resulting from the reconstruction of optimal investment decisions along the optimal investment trajectory according to (4.64). Thus, the general notion underlying our methodology is to be maximally invested in the investment trajectories ordered according to optimality.

Q trajectories, Diversification for a Subset of Times and Asynchronous Trading

Let us define a subset of trading sampling times as $\mathcal{T}^{(q)} \subseteq \{0, 1, \ldots, N_t\}$. A simple example is $\mathcal{T}^{(q)} = \{1\},\ \forall q = 0, 1, \ldots, Q-1$.

For enforcement of diversification in form of Q trajectories, diversification for any subset of trading times and asynchronous trading, the sets of admissible states are initialized as

$$\mathcal{Z}_0^{(q)} = \left\{ z_0 : z_0 = \begin{bmatrix} 0 & 0 & 0 & m_0^{0,q} & 0 & m_0^{0,q} & 0 & 0 \end{bmatrix}^{\mathsf{T}} \right\}, \tag{4.67}$$

for all $q = 0, \ldots, Q-1$.

For unconstrained trading frequency along an investment trajectory and $t > 0$, the sets of admissible states are consequently generated according to

$$\begin{aligned} \mathcal{Z}_t^{(q)} = \Bigg\{ & z_t : \max_{j_t \in \mathcal{J}_{t-1}^{z_t}} \{w_t^0\},\ \forall i_t \in \mathcal{I} \text{ if } t \notin \mathcal{T}^{(q)}, \text{ or ...} \\ & \forall i_t \in \mathcal{I} \backslash \left(\cup \left\{ i_t : i_t = i_t^{(r),\star}, z_t^{(r),\star} \in \mathcal{Z}_t^{(r),\star} \right\}_{r=0}^{q-1} \right) \text{ if } t \in \mathcal{T}^{(q)} \Bigg\}, \end{aligned} \tag{4.68}$$

with $q = 0, 1, \ldots, Q-1$ and whereby $z_t^{(r),\star} \in \mathcal{Z}_t^{(r)}$ denotes the optimal state at time t associated with investment trajectory r.

For the case of at most K admissible trades along any investment trajectory and $t > 0$, the sets of admissible states are consequently generated according to

$$\begin{aligned} \mathcal{Z}_t^{(q)} = \Bigg\{ & z_t : \max_{j_t \in \mathcal{J}_{t-1}^{z_t}} \{w_t^0\},\ \forall k_t < K \text{ and unique, and } \forall i_t \in \mathcal{I} \text{ if } t \notin \mathcal{T}^{(q)}, \\ & \text{or } \forall k_t < K \text{ and unique, and} \\ & \forall i_t \in \mathcal{I} \backslash \left(\cup \left\{ i_t : i_t = i_t^{(r),\star}, z_t^{(r),\star} \in \mathcal{Z}_t^{(r),\star} \right\}_{r=0}^{q-1} \right) \text{ if } t \in \mathcal{T}^{(q)} \Bigg\}, \end{aligned} \tag{4.69}$$

with $q = 0, 1, \ldots, Q-1$

For the case of enforcing a waiting period after each executed trade along any investment trajectory and $t > 0$, the sets of admissible states

are consequently generated according

$$\mathcal{Z}_t^{(q)} = \Bigg\{ z_t : \max_{j_t \in \mathcal{J}_{t-1}^{z_t}} \{w_t^0\},\ \forall d_t < D \text{ and unique, and } \forall i_t \in \mathcal{I} \text{ if } t \notin \mathcal{T}^{(q)},$$
$$\text{or } \forall d_t < D \text{ and unique, and}$$
$$\forall i_t \in \mathcal{I} \backslash \left(\cup \left\{ i_t : i_t = i_t^{(r),\star}, z_t^{(r),\star} \in \mathcal{Z}_t^{(r),\star} \right\}_{r=0}^{q-1} \right) \text{ if } t \in \mathcal{T}^{(q)} \Bigg\}, \tag{4.70}$$

with $q = 0, 1, \ldots, Q-1$.

Q trajectories, Diversification for all Times and Synchronous Trading

Let us define a subset of trading sampling times as $\mathcal{T} \subseteq \{0, 1, \ldots, N_t\}$. This subset may, for example, indicate the sampling time at which trades were executed along the optimal investment trajectory associated with $\mathcal{Z}_t^{(0),\star}$:

$$\mathcal{T} = \{t : i_t^{(0),\star} \neq j_t^{(0),\star},\ z_t^{(0),\star} \in \mathcal{Z}_t^{(0),\star},\ \forall t = 1, \ldots, N_t\}.$$

The set of a admissible states is initialized as in (4.67). Then, for unconstrained trading frequency along an investment trajectory and $t > 0$, the sets of admissible states are consequently generated according

$$\mathcal{Z}_t^{(q)} = \Bigg\{ z_t : z_t = z_{t-1} \text{ if } t \notin \mathcal{T}, \text{ or } z_t \text{ s.t. } \ldots$$
$$\max_{j_t \in \mathcal{J}_{t-1}^{z_t}} \{w_t^0\},\ \forall i_t \in \mathcal{I} \backslash \left(\cup \left\{ i_t : i_t = i_t^{(r),\star}, z_t^{(r),\star} \in \mathcal{Z}_t^{(r),\star} \right\}_{r=0}^{q-1} \right) \text{ if } t \in \mathcal{T} \Bigg\}, \tag{4.71}$$

with $q = 0, 1, \ldots, Q-1$.

The case of at most K admissible trades along any investment trajectory as well as the case of enforcing a waiting period after each executed trade along any investment trajectory can then be defined analogously.

Remarks and Relevant Quantities for Interpretation

In general, a mean for increasing computational efficiency is to reduce branches or nodes from the transition graph. This can be done based on

heuristics (branch and bound) that early discard investment trajectories not having any potential to result in an optimal investment trajectory.

The presented framework can also be used to analyze alternative optimization criteria such as determining a worst-case investment trajectory (*pessimization*) or the tracking of a target reference return trajectory.

Several quantities can be used to interpret results. We therefore first define the *total return* (measured in percent) as

$$r_{N_t}^{\text{tot},(q)} = 100\frac{w_{N_t}^0 - w_0^0}{w_0^0}, \quad \forall q \in \mathcal{Q}.$$

Similarly, we define the return at time t as $r_t^{\text{tot},(q)}$, $\forall q \in \mathcal{Q}$. We further report the total number of conducted trades as $K_{N_t}^{\text{tot}}$. The minimal timespan between any two trades within time-frame $t \in \mathcal{N}_t = \{0, 1, \ldots, N_t\}$ shall be denoted by $D_{N_t}^{\min}$.

In addition, the average, minimal and maximal percentage gain per conducted *non-currency* asset-trade is of our interest. Stating the quantities with respect to our reference currency (EUR), we therefore first define the set

$$\begin{aligned}\Delta\mathcal{G}^{(q)} = \Big\{ & 100\frac{w_\tau^0 - w_\eta^0}{w_\eta^0} : \text{with } \tau \text{ s.t. } \tau = t-1,\ \bar{i} \in \mathcal{I}_{N_a},\ i_{t-1} \neq \bar{i},\ i_t = \bar{i}, \\ & \text{with } \eta \text{ s.t. } \eta = t,\ \bar{i} \in \mathcal{I}_{N_a},\ i_t = \bar{i},\ i_{t+1} \neq \bar{i}, \text{ and } \tau > \eta,\ z_t \in \mathcal{Z}_t^{(q),\star}, \\ & \forall t \in \mathcal{N}_t,\ \forall q \in \mathcal{Q} \Big\},\end{aligned}$$

whereby $\bar{i}$ identifies an asset of interest. The average, minimum and maximum shall then be denoted by $\text{avg}(\Delta\mathcal{G}^{(q)})$, $\min(\Delta\mathcal{G}^{(q)})$ and $\max(\Delta\mathcal{G}^{(q)})$, respectively. The associated trading times are summarized in

$$\begin{aligned}\Delta\mathcal{T}^{(q)} = \Big\{ & \tau - \eta : \text{with } \tau \text{ s.t. } \tau = t-1,\ \bar{i} \in \mathcal{I}_{N_a},\ i_{t-1} \neq \bar{i},\ i_t = \bar{i}, \\ & \text{with } \eta \text{ s.t. } \eta = t, \bar{i} \in \mathcal{I}_{N_a},\ i_t = \bar{i},\ i_{t+1} \neq \bar{i}, \text{ and } \tau > \eta,\ z_t \in \mathcal{Z}_t^{(q),\star}, \\ & \forall t \in \mathcal{N}_t,\ \forall q \in \mathcal{Q} \Big\},\end{aligned}$$

with corresponding $\text{avg}(\Delta\mathcal{T}^{(q)})$, $\min(\Delta\mathcal{T}^{(q)})$ and $\max(\Delta\mathcal{T}^{(q)})$ defined accordingly.

Table 17: Example 1. Identification of currencies and assets under consideration. Each currency is associated with a foreign exchange rate with respect to EUR. The currency in which an asset i is traded is denoted by $c(i)$. The reference currency for the Nasdaq-100 is USD.

i	`finance.yahoo`-symbol	Interpretation	$c(i)$
0	–	EUR	0
1	`EURUSD=x`	USD	1
2	`^ NDX`	Nasdaq-100	1

Then, we can partition quantities of interest into two groups: overall performance measures and, secondly, quantities associated with non-currency asset holdings along an optimal q-trajectory. We therefore compactly summarize results in *evaluation vectors* and *evaluation matrices*

$$e^{(q)} = \begin{bmatrix} r_{N_t}^{\text{tot},(q)} & K_{N_t}^{\text{tot},(q)} & D_{N_t}^{\text{min},(q)} \end{bmatrix}, \quad \forall q \in \mathcal{Q}, \tag{4.72}$$

$$E^{(q)} = \begin{bmatrix} \text{avg}(\Delta\mathcal{G}^{(q)}) & \min(\Delta\mathcal{G}^{(q)}) & \max(\Delta\mathcal{G}^{(q)}) \\ \text{avg}(\Delta\mathcal{T}^{(q)}) & \min(\Delta\mathcal{T}^{(q)}) & \max(\Delta\mathcal{T}^{(q)}) \end{bmatrix}, \quad \forall q \in \mathcal{Q}. \tag{4.73}$$

4.3.6 Numerical Examples

To evaluate results, we consider three numerical examples. For all examples, a time horizon of one year is chosen. The sampling time is selected as one day. The influence of different levels of transaction costs on optimal trading decisions and quantitative performance measures are of main interest. Adjusted closing prices (accounting for all corporate actions such as stock splits and dividend payments) of both foreign exchange rates and stock indices are retrieved from `finance.yahoo.com`. Yearly data could not be retrieved for all stock indices of interest. Therefore, for some stock indices, we obtained data of an emulating ETF instead. For Example 3, we directly used stock price data. For uniformality, we then normalized all non-currency assets to value 100 in the

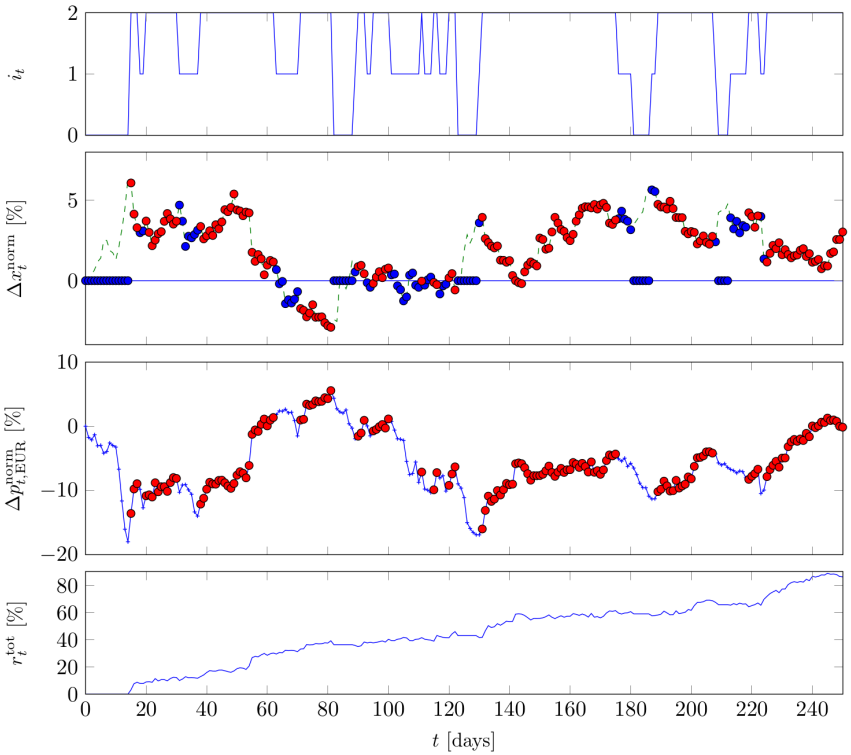


Figure 89: Example 1, the unconstrained trading case for proportional transaction costs of 1%. See also Table 18.

corresponding currency at time $t = 0$. See Tables 17 and 19 for details and the currencies associated with the assets.

The first example treats optimal trading of EUR, USD and the Nasdaq-100. This scenario is mainly selected to analyze currency effects. We here do not employ a diversification constraint, i.e., we have $Q = 1$. The second example treats optimal trading of 16 different currencies and 15 different non-currency assets. A diversification constraint is employed with $Q = 3$. The third example compares achievable performances for an exemplary downtrending and another uptrending stock when optimally trading a posteriori.

Table 18: Results of Example 1, where $\mathbf{e}^{(\mathbf{0})}$ and $\mathbf{E}^{(\mathbf{0})}$ are stacked atop each other for the different trading strategies; except for the Buy-and-Hold strategy, where only $\mathbf{e}^{(\mathbf{0})}$ is reported.

	$\epsilon = 0$	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$
Buy-and-Hold	$\begin{bmatrix}\mathbf{-0.2} & 1 & 250\end{bmatrix}$	$\begin{bmatrix}\mathbf{-1.2} & 1 & 250\end{bmatrix}$	$\begin{bmatrix}\mathbf{-2.2} & 1 & 250\end{bmatrix}$	$\begin{bmatrix}\mathbf{-4.2} & 1 & 250\end{bmatrix}$
Unconstrained	$\begin{bmatrix}\mathbf{330.0} & 165 & 1\end{bmatrix}$ $\begin{bmatrix}2.0 & 0.1 & 11.0\\ 1.8 & 1 & 5\end{bmatrix}$	$\begin{bmatrix}\mathbf{134.6} & 59 & 1\end{bmatrix}$ $\begin{bmatrix}3.9 & 0.9 & 11.8\\ 5.5 & 1 & 21\end{bmatrix}$	$\begin{bmatrix}\mathbf{86.2} & 31 & 1\end{bmatrix}$ $\begin{bmatrix}5.9 & 0.4 & 16.6\\ 12.2 & 1 & 45\end{bmatrix}$	$\begin{bmatrix}\mathbf{50.1} & 14 & 1\end{bmatrix}$ $\begin{bmatrix}8.2 & 3.1 & 15.5\\ 20.0 & 3 & 45\end{bmatrix}$
≤ 12 trades	$\begin{bmatrix}\mathbf{106.8} & 12 & 6\end{bmatrix}$ $\begin{bmatrix}11.9 & 7.1 & 17.8\\ 23.7 & 11 & 45\end{bmatrix}$	$\begin{bmatrix}\mathbf{87.9} & 12 & 6\end{bmatrix}$ $\begin{bmatrix}11.0 & 6.6 & 17.2\\ 23.7 & 11 & 45\end{bmatrix}$	$\begin{bmatrix}\mathbf{72.5} & 12 & 5\end{bmatrix}$ $\begin{bmatrix}10.1 & 5.8 & 16.6\\ 23.8 & 11 & 45\end{bmatrix}$	$\begin{bmatrix}\mathbf{49.6} & 12 & 1\end{bmatrix}$ $\begin{bmatrix}9.2 & 5.0 & 15.5\\ 23.7 & 11 & 45\end{bmatrix}$
≥ 10 days waiting	$\begin{bmatrix}\mathbf{114.2} & 18 & 10\end{bmatrix}$ $\begin{bmatrix}9.1 & 2.7 & 16.6\\ 15.3 & 10 & 21\end{bmatrix}$	$\begin{bmatrix}\mathbf{84.8} & 17 & 10\end{bmatrix}$ $\begin{bmatrix}9.2 & 2.7 & 16.0\\ 16.1 & 11 & 21\end{bmatrix}$	$\begin{bmatrix}\mathbf{68.4} & 13 & 10\end{bmatrix}$ $\begin{bmatrix}9.8 & 6.0 & 15.4\\ 22.3 & 11 & 45\end{bmatrix}$	$\begin{bmatrix}\mathbf{46.5} & 10 & 11\end{bmatrix}$ $\begin{bmatrix}9.3 & 5.9 & 15.5\\ 25.4 & 12 & 45\end{bmatrix}$

Example 1: EUR, USD and Nasdaq-100

The results of experiments for numerical Example 1 are summarized in Table 18. We consider different levels of transaction costs with variable proportional cost but constant fixed cost. The evaluation quantities $\mathbf{e}^{(\mathbf{0})}$ and $\mathbf{E}^{(\mathbf{0})}$ are stacked atop each other for the different trading strategies; except for the Buy-and-Hold strategy, where only $\mathbf{e}^{(\mathbf{0})}$ is reported. We assume proportional costs (indicated in $\%$) to be the same for buying and selling for both foreign exchange and asset trading, i.e., $\epsilon = \epsilon_{\text{buy}} = \epsilon_{\text{sell}}$. For $\epsilon = 0$, we also set $\beta = 0$. For all other cases, we set $\beta = 50$. Total returns ($r_{N_t}^{\text{tot},(q)}$) are printed bold for emphasis. The time-span of interest is August 5, 2015 until August 3, 2016. It comprises 251 potential trading days.

Several observations can be made with respect to the results of Table 18. First, eventhough only two currencies, EUR and USD, i.e., $i \in \mathcal{I}_{N_c} = \{0, 1\}$, and one non-currency asset, i.e., $i \in \mathcal{I}_{N_a} = \{2\}$, are traded *long-only*, remarkable profits can be earned when optimally trading a posteriori. Even in case of (high) transaction costs with a proportional rate of 2%, the profits significantly outperform a one-year Buy-and-Hold strat-

egy. Second, the influence of different levels of transaction costs is impressive. Specifically for the *unconstrained* trading strategy with respect to returns, trading frequency and percentage gains (average, minimum and maximum) upon which the non-currency asset is traded. Third, while the total return drops with increasing transaction cost levels, the remaining evaluation quantities remain approximately constant for the K-trades strategy (here $K = 12$, i.e., 12 trades per year or one per month). Fourth, the results associated with the percentage gains upon which the non-currency asset is traded were unexpected. Intuitively, they were thought to be higher. The same holds for optimal time periods between any two trades. Results from Example 1 encourage frequent trading. For example, for the case with a waiting constraint, trading is encouraged upon percentage gains of on average slightly less than 10% for all four levels of transaction costs.

Figure 89 exemplatorily visualizes results. In order to compactly display multiple foreign exchange rates, we normalize w.r.t. the initial value at $t = 0$, see the subplot with label Δx_t^{norm}. For reference currency EUR, we set $\Delta x_t^{\text{norm}} = 0,\ \forall t = 0, 1, \ldots, N_t$. Analogously, we normalize non-currency prices and additionally take currency effects into account by first converting prices to currency EUR, see the subplot with label $\Delta p_{t,\text{EUR}}^{\text{norm}}$. For a specific optimal investment trajectory, at every time t, an investment in exactly one currency or non-currency asset is taken. Being invested in a non-currency asset is indicated by red balls in aforementioned figures. Since non-currency assets are associated with a specific currency we also label them correspondingly with red balls. In contrast, an explicit investment in a currency is emphasized by blue balls.

It is striking that despite absence of clear trends in both the EURUSD-foreign exchange rate and the Nasdaq-100 stock index, significant profits can be made when optimally trading–even when employing a *long-only* strategy. The largest increases in return rates in currency EUR are achieved when the asset is increasing in value while the foreign exchange rate with reference Euro is decreasing. Investments in USD are optimal when the EURUSD-foreign exchange rate is trending down and the Nasdaq-100 is decreasing likewise. Investments in EUR are in general optimal

when the EURUSD-foreign exchange rate is trending up and the Nasdaq-100 is trending down.

Example 2: Global Investing and a Diversification Constraint

We distinguish two cases: synchronous and asynchronous trading. Quantitative results are summarized in Tables 20 and 21, respectively. The results for all Q trajectories are reported. The "Summary"-sections in Tables 20 and 21 report the sum of returns of all Q trajectories. Results are further visualized in Figure 90. The black-dashed horizontal line in the corresponding top subplots denotes $N_c = 16$ to distinguish currency and non-currency asset investments.

For performance comparison, we consider a Buy-and-Hold strategy, whereby an asset is bought initially and then held. The most performant non-currency assets from Table 19 for the time frame of interest were, in order, the IBOVESPA (BRA), the Dow Jones Russia GDR (RUS) and the S&P 500 (USA). Associated returns are reported in Table 20 and 21, where we attribute the IBOVESPA to $q = 0$ and the other two assets to $q = 1$ and $q = 2$, respectively.

Interpretation of results is in line with Experiment 1. In particular, the influence of transaction costs and the encouragement of frequent trading upon relatively small percentage gains can be observed likewise.

A remark about computational complexity needs to be made. The total number of states, $N_z(N_t)$, without consideration of any heuristics is 6139, 71611 and 59905 for the three cases (unconstrained, constraint of at most $K = 12$ trades, and constraint of waiting at least $D = 10$ days between any two trades). These numbers can be computed according to the formulas stated in Section 4.3.4. Then, applying the heuristics from Section 4.3.4 to the given `finance.yahoo`-data trajectories, we measured (to give one example) $N_z(N_t) = 65238$ and $N_z(N_t) = 33161$ for the latter two cases, $q = 0$ and $\epsilon = 0$. Similar results are obtained for the other transaction cost levels and the other q-trajectories, resulting in overall computation times (for all $q = 0, 1, 2$) in the tens of minutes. In contrast, for the unconstrained case, overall computation times for the generation of all $Q = 3$ transition graphs were on average only slightly more than

Table 19: Example 2. Identification of 16 currencies and 15 assets. Each currency is associated with a foreign exchange rate with respect to EUR. The currency in which an asset i is traded is denoted by $c(i)$.

i	`finance.yahoo`-symbol	Interpretation	$c(i)$
0	–	EUR	0
1	`EURUSD=x`	USD	1
2	`EURJPY=x`	JPY	2
3	`EURGBP=x`	GBP	3
4	`EURCHF=x`	CHF	4
5	`EURCNY=x`	CNY	5
6	`EURDKK=x`	DKK	6
7	`EURHKD=x`	HKD	7
8	`EURNOK=x`	NOK	8
9	`EURRUB=x`	RUB	9
10	`EURBRL=x`	BRL	10
11	`EURAUD=x`	AUD	11
12	`EURCAD=x`	CAD	12
13	`EURTRY=x`	TRY	13
14	`EURZAR=x`	ZAR	14
15	`EURINR=x`	INR	15
16	`^GDAXI`	DAX (GER)	0
17	`FTSEMIB.MI`	FTSEMIB (ITA)	0
18	`^OSEAX`	OSEAX (NOR)	8
19	`CSSMI.SW`	SMI (SUI)	4
20	`^NDX`	Nasdaq-100 (USA)	1
21	`^GSPC`	S&P 500 (USA)	1
22	`^N225`	NIKKEI 225 (JPN)	2
23	`^HSI`	Hang-Seng (HKG)	7
24	`^BVSP`	IBOVESPA (BRA)	10
25	`^AORD`	All Ordinaries (AUS)	11
26	`^GSPTSE`	S&P/TSX (CAN)	12
27	`AFS.PA`	FTSE/JSE (RSA)	14
28	`RUS.PA`	Dow Jones Russia (RUS)	0
29	`INR.PA`	MSCI India (IND)	0
30	`TUR.PA`	Dow Jones Turkey (TUR)	0

Table 20: Summary of quantitative results of Example 2 for the first case: time-asynchronous trading with diversification for all times.

		$\epsilon = 0$	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$
$q = 0$	Buy-and-Hold	[**18.2** 1 199]	[**17.0** 1 199]	[**15.9** 1 199]	[**13.6** 1 199]
	Unconstrained	[**17450.5** 184 1] [3.3 0.1 19.5] [1.1 1 3]	[**3092.0** 126 1] [4.3 0.2 21.9] [1.7 1 5]	[**1099.5** 95 1] [5.9 0.7 24.7] [2.7 1 8]	[**354.5** 50 1] [8.6 1.8 26.2] [5.0 1 21]
	$\leq$ 12 trades	[**453.9** 12 3] [19.0 5.6 56.7] [13.8 3 55]	[**377.9** 12 3] [12.0 7.2 17.8] [22.5 11 45]	[**326.4** 12 2] [20.8 5.4 54] [18.4 3 49]	[**258.7** 12 2] [22.3 5.4 50.1] [19.0 3 46]
	$\geq$ 10 days waiting	[**440.8** 17 10] [12.9 3.8 33.5] [11.7 5 16]	[**341.2** 16 10] [11.5 2.1 31.4] [11.7 5 16]	[**269.3** 15 10] [11.1 1.8 29.5] [12.4 5 16]	[**186.6** 12 10] [17.9 12.1 26.7] [18.5 11 36]
$q = 1$	Buy-and-Hold	[**3.3** 1 199]	[**2.8** 1 199]	[**2.3** 1 199]	[**1.3** 1 199]
	Unconstrained	[**3671.6** 190 1] [2.4 0.02 7.5] [1.0 1 2]	[**769.0** 123 1] [3.3 0.4 9.0] [1.8 1 6]	[**378.5** 76 1] [4.8 1.2 9.0] [3.3 1 9]	[**173.2** 38 1] [8.0 2.8 22.5] [7.0 2 19]
	$\leq$ 12 trades	[**227.0** 12 1] [12.8 0.5 36.0] [15.5 1 43]	[**212.3** 12 1] [14.5 2.6 34.5] [16.2 1 43]	[**173.0** 12 4] [13.8 2.6 33.2] [18.3 7 43]	[**129.6** 12 3] [18.7 9.5 30.5] [23.7 7 43]
	$\geq$ 10 days waiting	[**267.5** 17 10] [10.4 2.0 18.4] [11.2 1 15]	[**200.5** 17 10] [9.8 3.8 17.2] [11.5 10 14]	[**153.2** 17 10] [9.2 1.5 14.5] [11.0 8 14]	[**113.3** 15 10] [9.1 1.2 22.1] [14.3 6 33]
$q = 2$	Buy-and-Hold	[**1.7** 1 199]	[**0.6** 1 199]	[**−0.4** 1 199]	[**−2.4** 1 199]
	Unconstrained	[**1882.1** 191 1] [2.0 2e − 14 6.9] [1.0 1 3]	[**459.7** 113 1] [3.0 0.1 11.4] [2.0 1 8]	[**229.3** 66 1] [4.9 1.2 12.0] [3.7 1 10]	[**102.9** 37 1] [6.4 2.9 10.0] [6.7 1 16]
	$\leq$ 12 trades	[**186.3** 12 1] [12.9 7.2 31.8] [16.6 1 67]	[**158.0** 12 1] [11.8 5.8 30.6] [15.3 1 49]	[**137.9** 12 1] [11.9 7.5 24.6] [18.0 5 37]	[**89.0** 12 3] [10.7 5.9 18.3] [18.0 8 36]
	$\geq$ 10 days waiting	[**215.2** 18 10] [9.2 3.3 13.9] [10.8 9 14]	[**164.1** 18 10] [7.5 2.5 12.4] [10.6 8 14]	[**116.2** 15 10] [9.0 4.9 12.3] [13.3 10 20]	[**70.7** 11 10] [9.7 5.8 18.3] [17.6 10 29]
Summary	Buy-and-Hold	**23.2**	**20.4**	**17.8**	**12.5**
	Unconstrained	**23004.2**	**4320.7**	**1707.3**	**630.6**
	$\leq$ 12 trades	**867.2**	**748.2**	**637.3**	**477.3**
	$\geq$ 10 days waiting	**923.5**	**705.8**	**538.7**	**370.6**

Table 21: Summary of quantitative results of Example 2 for the second case: time-synchronized trading with diversification for all times.

		$\epsilon = 0$	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$
$q = 0$	Buy-and-Hold	[**18.2** 1 199]	[**17.0** 1 199]	[**15.9** 1 199]	[**13.6** 1 199]
	Unconstrained	[**17450.5** 184 1] [3.3 0.1 19.5] [1.1 1 3]	[**3092.0** 126 1] [4.3 0.2 21.9] [1.7 1 5]	[**1099.5** 95 1] [5.9 0.7 24.7] [2.7 1 8]	[**354.5** 50 1] [8.6 1.8 26.2] [5.0 1 21]
	$\leq$ 12 trades	[**454.3** 12 2] [19.0 7.0 56.7] [13.8 2 55]	[**377.9** 12 3] [19.0 5.4 54.3] [15.3 3 55]	[**326.4** 12 2] [20.8 5.4 54.0] [18.4 3 49]	[**258.7** 12 2] [22.3 5.4 50.1] [19.0 3 46]
	$\geq$ 10 days waiting	[**440.8** 17 10] [12.9 3.8 33.5] [11.7 5 16]	[**341.2** 16 10] [11.5 2.1 31.4] [11.7 5 16]	[**269.3** 15 10] [11.1 1.8 29.5] [12.4 5 16]	[**186.6** 12 10] [17.9 12.1 26.7] [18.5 11 36]
$q = 1$	Buy-and-Hold	[**1.7** 1 199]	[**0.6** 1 199]	[**−0.4** 1 199]	[**−2.4** 1 199]
	Unconstrained	[**3009.4** 177 1] [2.4 0.02 7.8] [1.1 1 3]	[**523.5** 89 1] [3.6 0.4 13.0] [2.5 1 8]	[**259.9** 46 1] [6.1 0.6 18.6] [5.8 1 17]	[**137.8** 24 1] [9.1 2.9 25.9] [10.3 2 23]
	$\leq$ 12 trades	[**142.8** 12 2] [8.9 5.0 18.0] [11.3 2 55]	[**112.7** 12 3] [8.2 0.8 16.8] [13.5 4 55]	[**87.7** 12 2] [9.8 5.4 19.7] [14.8 4 49]	[**67.3** 12 2] [11.1 3.9 22.4] [15.3 4 46]
	$\geq$ 10 days waiting	[**179.1** 17 10] [7.6 0.7 15.7] [11.7 5 16]	[**135.9** 14 10] [8.1 0.8 15.1] [12.8 5 24]	[**116.0** 12 10] [9.0 2.1 23.4] [16.6 10 41]	[**73.2** 9 10] [11.3 6.0 14.8] [19.8 11 32]
$q = 2$	Buy-and-Hold	[**3.3** 1 250]	[**2.1** 1 250]	[**1.1** 1 250]	[**−0.9** 1 250]
	Unconstrained	[**1640.8** 177 1] [2.1 0.02 10.6] [1.1 1 4]	[**342.9** 91 1] [3.2 0.1 9.5] [2.4 1 6]	[**170.3** 45 1] [5.8 0.4 11.4] [6.1 1 21]	[**87.9** 20 1] [7.9 3.9 11.7] [11.0 3 22]
	$\leq$ 12 trades	[**114.8** 12 2] [8.2 4.6 14.6] [11.6 2 55]	[**92.2** 11 3] [9.4 0.6 24.6] [15.7 3 59]	[**74.4** 11 2] [10.3 1.8 28.2] [18.3 3 53]	[**50.0** 7 6] [15.9 3.8 25.6] [36.0 6 53]
	$\geq$ 10 days waiting	[**134.2** 16 10] [6.4 0.3 11.7] [12.4 10 16]	[**102.9** 13 10] [8.4 0.5 12.2] [17.3 11 33]	[**87.8** 12 11] [7.3 4.5 10.1] [14.7 13 21]	[**52.8** 7 10] [13.9 8.1 19.3] [26.7 25 32]
Summary	Buy-and-Hold	**23.2**	**20.4**	**17.8**	**12.5**
	Unconstrained	**22100.7**	**3958.4**	**1529.7**	**580.2**
	$\leq$ 12 trades	**711.9**	**582.8**	**488.5**	**376.0**
	$\geq$ 10 days waiting	**754.1**	**580.0**	**473.1**	**312.6**

Table 22: Summary of quantitative resultus of Example 3. Comparison of a downtrending and uptrending stock for time period August 10, 2015 until August 8, 2016. The exemplary downtrending stock is of Deutsche Bank AG (`finance.yahoo`-symbol: `DBK.DE`). The exemplary uptrending stock is of Adidas AG (`finance.yahoo`-symbol: `ADS.DE`).

DKB.DE	$\epsilon = 1$
Buy-and-Hold	$\begin{bmatrix}\mathbf{-61.4} & 1 & 260\end{bmatrix}$
Unconstrained	$\begin{bmatrix}\mathbf{382.8} & 51 & 1\end{bmatrix}$ $\begin{bmatrix}7.5 & 1.5 & 21.8\\ 4.2 & 1 & 11\end{bmatrix}$

ADS.DE	$\epsilon = 1$
Buy-and-Hold	$\begin{bmatrix}\mathbf{95.6} & 1 & 260\end{bmatrix}$
Unconstrained	$\begin{bmatrix}\mathbf{249.4} & 43 & 1\end{bmatrix}$ $\begin{bmatrix}7.0 & 1.0 & 20.1\\ 7.8 & 2 & 24\end{bmatrix}$

10 seconds, thereby making the unconstrained case much more suitable for fast analysis of sets of multiple assets and foreign exchange rate trajectories. Secondly, the trajectories for $q = 0$ are identical for both time-asynchronous and -synchronous trading. However, for the remaining investment trajectories with $q > 0$, the number of states is much lower for time-synchronous trading in comparison to the asynchronous case. For time-synchronous trading, $Q = 3$ and a bound on the total admissible number of trades, the total number of states is 63803 for $q = 0$, but 39713 and 23549 for $q = 1$ and $q = 2$, respectively. All numerical experiments throughout this section were conducted on a laptop running Ubuntu 14.04 equipped with an Intel Core i7 CPU @ 2.80GHz×8, 15.6 GB of memory, and using Python 2.7.

Example 3: A Downtrending and an Uptrending Stock

The final example compares achievable performances for an examplary downtrending and an an uptrending stock. The exemplary downtrending stock is of Deutsche Bank AG (`DKB.DE`). The exemplary uptrending stock is of Adidas AG (`ADS.DE`). Both stocks are listed in the German stock index (DAX). The time-frame considered is August 10, 2015 until August 8, 2016. There are 260 potential trading days. Both stocks are traded in currency EUR. We thus find optimal investment trajecto-

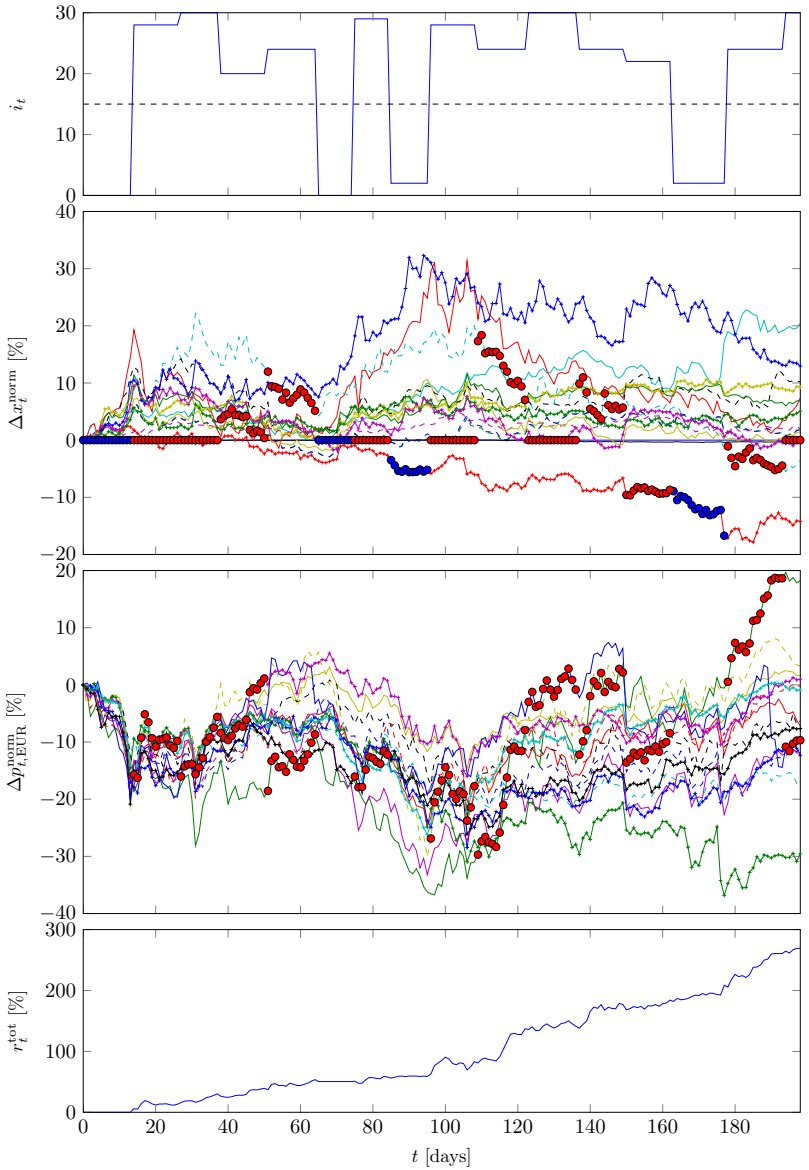


Figure 90: Example 2. The results for $q = 0$, a waiting period of at least $D = 10$ days between two trades and transaction cost level $\epsilon = 1$. For $q = 0$, the results for asynchronous and synchronous trading are identical.

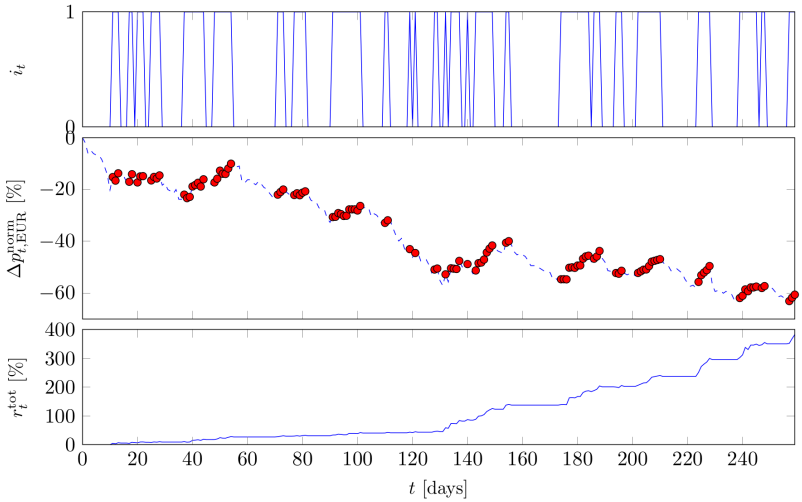


Figure 91: The downtrending stock of Example 3 for the unconstrained trading case in case of proportional transaction costs of 1%. The exemplary downtrending stock is of Deutsche Bank AG (`finance.yahoo`-symbol: `DKB.DE`). See also Table 22.

ries when a) trading `DKB.DE` and EUR, and b) trading `ADS.DE` and EUR. We assume propotional costs of 1% identical for buying and selling, i.e., $\epsilon = \epsilon_{\text{buy}} = \epsilon_{\text{sell}}$. We set $\beta = 50$. Results are summarized in Table 22 and Figures 91 and 92.

Unexpectedly and remarkably, the optimal investment trajectory and its associated yearly return for the downtrending stock is higher than for the optimal investment trajectory corresponding to the uptrending trend: 382.8% vs. 249.2%. Note that the corresponding Buy-and-Hold returns are −61.4% and 95.6%. While overall downtrending, the price of `DKB.DE` indicates temporary steep price increases with occur mostly towards the second half of the time-period of interest. This implies stronger return growth due to already exponentially amplified available value hitherto (instead of the initial m_0). Naturally, without a posteriori knowledge of price evolutions, an uptrending stock such as `ADS.DE` offers the advantage that missing the right selling dates is less important. Interestingly,

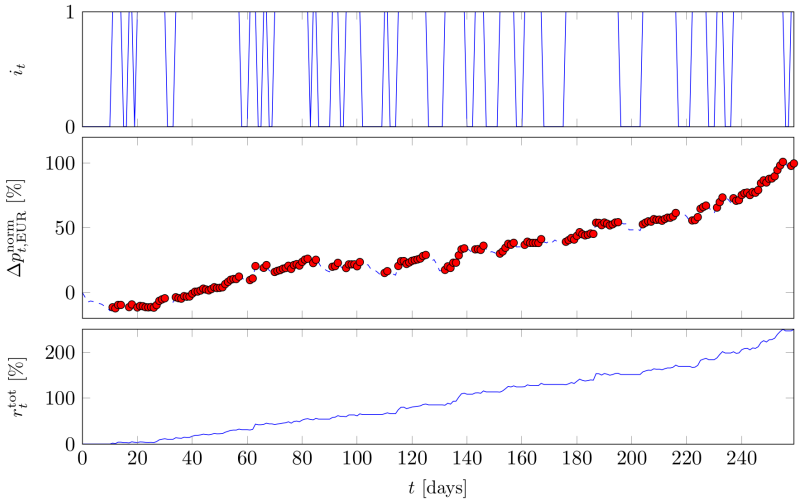


Figure 92: The uptrending stock of Example 3 for the unconstrained trading case in case of proportional transaction costs of 1%. The exemplary uptrending stock is of Adidas AG (`finance.yahoo`-symbol: `ADS.DE`). See also Table 22.

both downtrending and uptrending are traded optimally upon similar short-term average price increases: 7.5% and 7%. Similarly, the optimal holding periods of the stocks are short, on average: 4.2 and 7.8 days, respectively.

4.3.7 Conclusion

We developed a method for a posteriori (historical) multi-stage optimal trading under transaction costs and a diversification constraint. Findings were evaluated on real-world data recorded for one year. We found that transaction cost levels are decisive for achievable performance and significantly influence optimal trading frequency. Quantitative results further indicated optimal trading upon occasion rather than on fixed trading intervals, and, dependent on transaction cost levels, upon single- to low double-digit percentage gains in value (with respect to the reference currency) exploiting short-term trends. Achievable returns for optimized

trading are overwhelming, and uncomparably outperforming Buy-and-Hold strategies. Naturally, these returns are very difficult to achieve without knowledge of future price and foreign exchange rate evolutions. Nevertheless, they indicate the *potential* of optimal trading and can provide valuable insights for the development of algorithmic trading and screening systems.

The developed methods can be used in multiple ways. First, for a set of given asset price evolutions, the most suitable or best combinations of technical indicators (such as, e.g., moving average crossovers, Williams %R, etc.) and chart pattern analysis techniques (such as, e.g., head and shoulders, wedge, etc.) can be determined that most closely replicate the trading recommendations of optimal investment trajectories. The task of the finding of combinations can also be posed as an optimization problem. Second, assuming multi-step predictions about future price or foreign exchange rate evolution can be made, the developed methods can be used to optimally assign investments. Third, the developed methods can be used in the design of an adaptive or self-learning trading recommendation system. Therefore, additional prediction methods have to be incorporated.

4.3.8 Hierarchical Controller Parametrization

There is one hierarchical layer. It is represented by any of the optimization problems from Section 4.3.4 and 4.3.5.

4.4 Single-Asset Stock Trading: Stochastic Model Predictive or Genetic?

This section summarizes [153]:

- M. Graf Plessen, and A. Bemporad, "Stock trading via feedback control: Stochastic model predictive or genetic?," originally presented as a poster at *XVIII Workshop on Quantitative Finance (QFW2017)*, to appear upon invitation in *Journal of Modern Accounting and Auditing*, available at *arXiv preprint arXiv: 1708.08857*, 2017.

A suitable feedback control structure for stock trading under proportional transaction costs is sought. Suitability refers to robustness and performance. Both are tested by considering different one-step ahead stock price prediction qualities, including the ideal case, correct prediction of the direction of change in daily stock prices and the worst-case. Feedback control structures are partitioned into two general classes: SMPC and genetic algorithms. For the former class, three controllers are discussed, whereby it is distinguished between two Markowitz- and one dynamic hedging-inspired SMPC formulation. For the latter class, five trading algorithms are disucssed, whereby it is distinguished between two different moving average (MA) based, two trading range (TR) based, and one strategy based on historical optimal (HistOpt) trajectories. The combinations of all of the eight controllers with five different one-step ahead prediction methods are backtested for daily trading of the 30 components of the DAX for the time period between November 27, 2015 and November 25, 2016.

4.4.1 Introduction

Within the context of performance-related asset trading, we distinuish between three general tasks: system identification (finding of cause and effect relations), scenario generation (future asset price predictions) and trade decision taking (control logic). This work focuses on the third task. For low-level trading mechanics and feedback control thereof we refer to [25]. For recent control theory-related research problems in finance

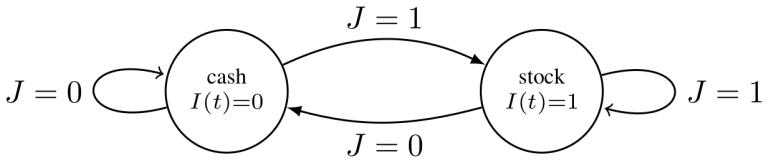


Figure 93: Visualization of the Markov decision process (MDP) when trading cash and a stock only.

associated with the control of order book dynamics, see [24]. This work is founded on [265] and [42]. The motivation for this work is the intention to extend a SMPC approach to multi-asset portfolio optimization for profit- and risk-related objectives. However, first, the suitability of SMPC needs to be verified and compared to alternative control strategies. Such a comparison is provided below. The main contribution of this work is analysis to find the most suitable general feedback control structure for stock trading out of two general and large classes: SMPC and genetic algorithms. Here, we refer to a genetic algorithm as any customized control method of arbitrary structure whose parameters are optimized through simulation using real-world data. We compare eight different stock trading algorithms that can be partitioned into the two aforementioned classes. For scenario generation, on which we rely all controllers, we assume five different one-step ahead stock price prediction methods. Their quality ranges from ideal (perfect price-ahead prediction) to totally off (wrong price rate sign-prediction at all sampling intervals). It is stressed that we explicitly do *not* consider multi-asset *portfolio optimization*, but instead concentrate on the trading of separate *single assets* for a given period of time. The used real-world data is drawn from the 30 components of the German stock market index DAX.

4.4.2 Transition Dynamics Modeling

Let time index t be associated with sampling time T_s such that all time instances of interest can be described as tT_s, whereby, throughout this

section, we have $T_s = 1$ day. Let us define the system state by

$$Z(t) = \begin{bmatrix} I(t) & M(t) & N(t) & W(t) \end{bmatrix}^\mathsf{T}, \tag{4.74}$$

with $I(t) \in \{0, 1\}$ indicating a cash- or stock-investment, respectively, $M(t) \in \mathbb{R}_+$ the current cash position (measured in currency €), $N(t) \in \mathbb{Z}_+$ the number of shares held, and $W(t) \in \mathbb{R}_+$ the current portfolio wealth. Thus, to analyze a suitable stock trading algorithm, we assume that *fractional* investments are not possible, and that investments are therefore always made either entirely in cash or a stock. Transition dynamics can then be modeled as a Markov decision process (MDP). Control variable $J(t) \in \{0, 1\}$ decides upon investment positions according to Figure 93. In general, we model transaction costs as non-convex with a fixed charge for any nonzero trade (fixed transaction costs) and a linear term scaling with the quantity traded (proportional transaction costs). Thus, at time $t-1$, the purchase of $N(t-1)$ shares of an asset results in $M(t) = M(t-1) - N(t-1)s(t-1)\left(1 + \epsilon_{\text{buy}}\right) - \beta_{\text{buy}}$, with $s(t)$ denoting asset (closing) price at time t. Similarly, for the selling of $N(t-1)$ assets we have $M(t) = M(t-1) + N(t-1)s(t-1)\left(1 - \epsilon_{\text{sell}}\right) - \beta_{\text{sell}}$. For the remainder of this section we assume no fixed transaction costs, i.e., $\beta_{\text{sell}} = \beta_{\text{buy}} = 0$. This simplification is done to directly adapt the convex problem formulations from [265] and for the dynamic hedging-inspired formulation proposed in Section 4.4.3. Fixed transaction costs, that render the problem non-convex, can be approached by iterative relaxation methods [257] or hybrid system theory. For wealth dynamics, we have $W(t) \in \left\{W(t-1), M(t), \tilde{W}(t)\right\}$, whereby $\tilde{W}(t) = \tilde{M}(t) + \tilde{N}(t)s(t)$ with $\tilde{M}(t)$ denoting the optimizer and $\tilde{N}(t)$ the optimal objective function value of

$$\max_{M(t)\geq 0} \left\{ N(t) \in \mathbb{Z}_+ : N(t) = \frac{M(t-1) - \beta_{\text{buy}} - M(t)}{s(t-1)\left(1 + \epsilon_{\text{buy}}\right)} \right\}.$$

Thus, given $M(t-1)$, we find the largest possible positive integer number of assets we can purchase under consideration of transaction costs. The smallest possible cash residual is $\tilde{M}(t) = 0$.

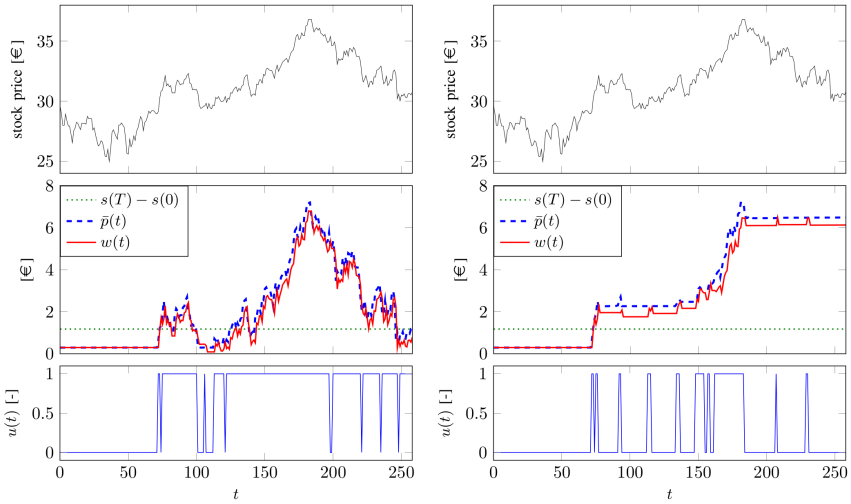


Figure 94: Visualization of the dynamic hedging-inspired SMPC formulation for stock trading. A noncausal setting with perfect $s(t+1)$ knowledge is assumed for illustration. Parameters for scenario generation and perturation noise are $(M, \sigma_{\text{pert}}) = (100, 0.3)$. (Left) Dynamic hedging result. (Right) Result of SMPC-DH. In both cases, (4.80) is solved. The difference is the generation of references $p^j(t+1),\ j = 1, \ldots, M$. The average is denoted by $\bar{p}(t) = \frac{1}{M} \sum_{j=1}^{M} p^j(t)$. The underlying stock (top frame) is of Vonovia SE (November 27, 2015 until November 25, 2016).

Within this work, the focus is on *unconstrained* trading frequency of *one* asset, i.e., trading is permitted on any two consecutive trading days, and confining cash and asset to be based on the *same* currency.

To summarize, at every trading interval t we conduct the following algorithm:

1. Read current $s(t)$ to update $W(t)$ and thus $Z(t)$.
2. Decide on $J(t) \in \{0, 1\}$.
3. Rebalance the portfolio according to $J(t)$.

All of the following two sections are concerned about the decision on $J(t) \in \{0, 1\}$ with fundamental objective profit maximization.

4.4.3 Stochastic Model Predictive Stock Trading

Let us discuss a relation between portfolio optimization and dynamic hedging using stochastic model predictive control (SMPC). For a financial institution, hedging a derivative contract implies to dynamically rebalance a so-called *replicating* portfolio of underlying assets at periodic intervals so that, at the expiration date of the contract, the value of the portfolio is as close as possible to the payoff value to pay to the customer. For the multiple asset replicating portfolio case, wealth dynamics $w(t)$ of the replicating portfolio can be defined as

$$w(t+1) = (1+r)\left(w(t) - \sum_{i=1}^{n} h_i(t)\right) + \sum_{i=1}^{n} b_i(t)u_i(t) \tag{4.75}$$

where $u_i(t)$ is the *fractional* quantity of asset i held at time t, $b_i(t) = s_i(t+1) - (1+r)s_i(t)$ is the excess return, i.e., how much the asset gains (or loses) with respect to the risk-free rate r over interval T_s, and transaction costs $h_i(t)$ are assumed to be proportional to the traded quantity of stock,

$$h_i(t) = \epsilon_i s_i(t)|u_i(t) - u_i(t-1)|, \tag{4.76}$$

with fixed quantity $\epsilon_i \geq 0$ depending on commissions on trading asset i, $\forall i = 1, \ldots, n$ (we assume no costs are applied on trasacting the risk-free asset). A standard option contract typically covers 100 shares. Thus, $u_i(t) = 1$ implies a portfolio such that at the end of the rebalancing interval, 100 shares of underlying asset i are held. For our setting, we here assume one asset only and drop subscripts accordingly. Furthermore, we set $\epsilon = \epsilon_{\text{buy}}$ and $\epsilon_{\text{sell}} = \epsilon_{\text{buy}}$. For the formulation of convex optimization problems, we introduce a *virtual* portfolio, constrain[6] $u(t) \in \{0, 1\}$, and then relate $I(t) = u(t)$.

Two Markowitz-inspired SMPCs

With regard of portfolio optimization, Markowitz [265] trades-off the mean (performance) and variance (risk) of the return. For our setting,

[6]The unconstrained case admitting $u(t) < 0$ would imply the possibility of *shortselling*.

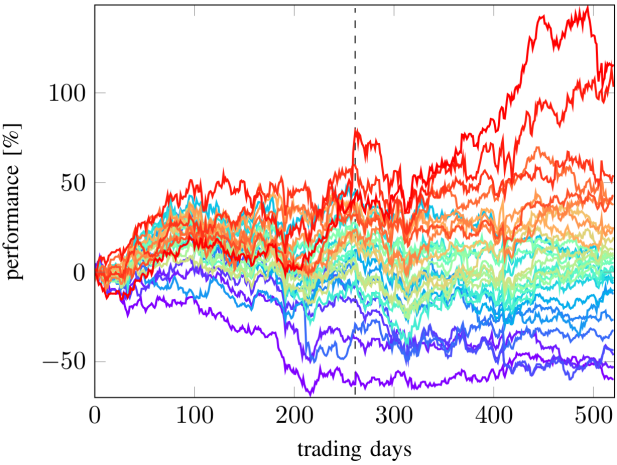


Figure 95: Normalized two-year evolution of all 30 components held in the DAX between November 28, 2014, and November 25, 2016. The data is partitioned by the black-dashed vertical line into training and evaluation data, respectively. Thus, $t = 0$ is initialized at trading day 261 (November 27, 2015).

this objective can be formulated as

$$\max_{u(t)\in\{0,1\}} \mathrm{E}[w(t+1)] - \frac{\alpha}{2}\mathrm{Var}[w(t+1)], \tag{4.77}$$

where α denotes the trade-off parameter. The decision of selecting $u(t)$ is largely dependent on $s(t+1)$, which is unknown at time t. Employing a SMPC approach, we therefore generate M scenarios for possible future prices $s^j(t+1)$ with corresponding probabilities $\pi^j,\ j = 1, \ldots, M$. Accordingly, we obtain $w^j(t+1)$, $\mathrm{E}[w(t+1)] = \sum_{j=1}^{M} \pi^j w^j(t+1)$ and $\mathrm{Var}[w(t+1)] = \mathrm{E}[w^2(t+1)] - \mathrm{E}^2[w(t+1)]$. We generate scenarios as

$$s^j(t+1) = \hat{s}(t+1) + \sigma_{\text{pert}}\eta^j(t),\ \eta^j(t) \sim \mathcal{N}(0,1), \tag{4.78}$$

where $\hat{s}(t+1)$ denotes our mean estimate of $s(t+1)$ and $\sigma_{\text{pert}} \geq 0$ is a tuning parameter to add *perturbation noise*. For final experiments we assume $M = 100$. Thus, our first SMPC-based controller solves (4.77) with scenario generation according (4.78). It is referred to as SMPC-M100. For

the *fractional* case relevant for dynamic option hedging (and specific for Δ-hedging [49]), the corresponding to (4.77) can be cast into a quadratic program (QP) by the introduction of two slack variables. For our case, we just evaluate the objective function for both $u(t) \in \{0, 1\}$ and therefore do not require a QP-solver.

In a second setting we assume $M = 1$ which implies $\text{Var}[w(t+1)] = 0$. In order to still maintain a possible knob to trade-off performance and risk, we define

$$\max_{u(t)\in\{0,1\}} \; \text{E}[w(t+1)] - \frac{\beta}{2}(u(t) - u(t-1))^2\sigma, \tag{4.79}$$

where β is a tuning parameter and σ was introduced to relate to data. Assuming the log-normal stock model, it is typically estimated as the *maximum likelihood* (ML) from $\mathcal{T}+1$ past stock prices using $\{\ln(\frac{s(t-\mathcal{T}+1)}{s(t-\mathcal{T})}), \ldots, \ln(\frac{s(t)}{s(t-1)})\}$. The higher β, the less frequent $u(t)$ is varied. Again, we solve (4.79) by evaluating both $u(t) \in \{0, 1\}$ instead of solving the more general QP. We refer to the resulting controller as SMPC-E+, eventhough, strictly it is not stochastic anymore since $M = 1$.

Dynamic Hedging-inspired SMPC

For option hedging, the objective is to typically minimize the so-called hedging error $e(T) = w(T) - p(T)$, where $w(T)$ and $p(T)$ denote replicating portfolio wealth and option price at expiration date T, respectively. Guiding notion of *dynamic* option hedging is to minimize the "tracking error", $e(t) = w(t) - p(t), \; \forall t = 0, \ldots, T$ for all possible asset price realization. In order to minimize under transaction costs, three different stochastic measures of the predicted hedging error are discussed in [42]. We here focus on the LP-MinMax formulation, minimizing the maximal hedging error resulting from scenario generation, i.e.,

$$\min_{u(t)\in\{0,1\}} \max_{j=1,\ldots,M} \; |w^j(t+1) - p^j(t+1)|. \tag{4.80}$$

Its benefit over the other two stochastic measures is independence from any trade-off parameter. Let us discuss how the framework (4.80) can

serve for stock trading. First, for hedging of a European call option, the analytical scheme to generate option price scenarios is $p^j(t+1) = (1+r)^{-(T-(t+1))}\max(s^j(T) - K_s, 0),\ j = 1, \ldots, M$. The option strike price is denoted by K_s. For our stock trading objective, we modify these "references". For (4.80), we therefore propose

$$p^j(t+1) = (1+r)^{-(T-(t+1))}\max(s^j(t+1) - s(0), l(t)), \tag{4.81}$$

with

$$l(t) = \begin{cases} \max(w(t),\ 0), & \text{if } w(t) > l(t), \\ l(t-1), & \text{otherwise,} \end{cases}$$

and initialize $l(0) = 0$. The corresponding dynamic hedging-inspired controller shall be referred to as SMPC-DH. Both the difference with respect to dynamic option hedging and the motivation for employing (4.81) for stock trading are visualized in Figure 94. Using (4.81) in combination with (4.80) can be interpreted as a *trailing stop-loss* strategy. Finally, we remark that the other two stochastic measures (QP-Var and LP-CVaR) from [42] can be employed likewise using (4.81).

4.4.4 Genetic Stock Trading

We discuss five genetic stock trading methods.

Two Moving Average-based Controllers

Let us define the moving average (MA) of a stock price as $s_{\text{MA}}(t+1) = \frac{1}{p_{\text{MA}}}\sum_{\tau=0}^{p_{\text{MA}}-1}\tilde{s}(t+1-\tau)$, with $\tilde{s}(t+1) = \hat{s}(t+1)$ and $\tilde{s}(t+1-\tau) = s(t+1-\tau),\ \forall\tau \geq 1$, and where p_{MA} is the moving average length parameter. The first MA-based controller, referred to as MA-Cross in the following, triggers a buy-signal ($J(t) = 1$) in case of a short-term MA coming from "below" (i.e., with a lower price one sampling time before) and crossing a long-term MA. Similarly, a sell-signal ($J(t) = 0$) is generated in case of the short-term MA crossing the long-term MA from "above" (i.e., with a higher price one sampling time before). The two parameters defining MA-lengths shall be denoted by $p_{\text{MA,s}}$ and $p_{\text{MA,l}}$.

The second MA-based controller, below referred to as MA-Sign, takes as input parameters p_{MA} and T_{MA}. It then computes $\Delta s_{\text{MA}}(t-\tau) = s_{\text{MA}}(t+1-\tau) - s_{\text{MA}}(t-\tau),\ \forall \tau = 0, 1, \ldots, T_{\text{MA}} - 2$. A buy-signal is generated if sign $(\Delta s_{\text{MA}}(t-\tau)) > 0,\ \forall \tau = 0, 1, \ldots, T_{\text{MA}} - 2$, a sell-signal otherwise, and where sign$(\cdot)$ denotes the sign-operator. The idea is to exploit price trends using constant-sign MA-slope rates for the past $T_{\text{MA}} - 1$ intervals. We refer to this second MA-based controller as MA-Sign.

Two Trading Range-based Controllers

Let us select a time window $[t - T_{\text{win}}, t]$ and partition it such that

$$T_{\text{win}} = K p_{\text{TR}} + \delta, \tag{4.82}$$

where $p_{\text{TR}} \in \mathbb{Z}_{++}$ is a parameter, $K \in \mathbb{Z}_{++}$, and $\delta \geq 0$ a corresponding residual of time-instances. We define interval-wise *local maxima* by

$$s_{\text{max}}^{(k)} = \max_{\tau \in [t - T_{\text{win}} + (k-1)K,\, t - T_{\text{win}} + kK]} s(\tau), \tag{4.83}$$

and the corresponding time arguments by $t_{\text{max}}^{(k)},\ \forall k = 1, \ldots, K$. Similarly, we derive *local minima* $s_{\text{min}}^{(k)}$ and $t_{\text{min}}^{(k)}$. Local minima and maxima are suitable to generate *trading ranges* (TR). We refer to our first TR-based controller as TR-Inside. It triggers trading signals as follows:

$$J(t) = \begin{cases} 0, & \text{if } \frac{|\hat{s}(t+1) - \hat{y}_{\text{max}}(t+1)|}{\hat{y}_{\text{max}}(t+1)} < \epsilon_{\text{TR}}, \\ 1, & \text{if } \frac{|\hat{s}(t+1) - \hat{y}_{\text{min}}(t+1)|}{\hat{y}_{\text{min}}(t+1)} < \epsilon_{\text{TR}}, \\ J(t-1), & \text{otherwise}, \end{cases}$$

where $\hat{y}_{\text{max}}(t+1) = (t + 1 - t_{\text{max}}^{(K)}) q_{\text{max}}(t+1) + s_{\text{max}}^{(K)}$, $q_{\text{max}}(t+1) = (s_{\text{max}}^{(K)} - s_{\text{max}}^{(K-1)}) / (t_{\text{max}}^{(K)} - t_{\text{max}}^{(K-1)})$, and analogously for $\hat{y}_{\text{min}}(t+1)$ and $q_{\text{min}}(t+1)$. Thus, buy(sell)-signals are triggered upon reaching the lower(upper) trading range affinely constructed based upon the last two local minima(maxima).

Our second TR-based controller is referred to as TR-Outside. It trig-

gers trading signals according to:

$$J(t) = \begin{cases} 1, & \text{if } \frac{\hat{s}(t+1)-\hat{y}_{\max}(t+1)}{\hat{y}_{\max}(t+1)} > \epsilon_{\text{TR}}, \\ 0, & \text{if } \frac{\hat{s}(t+1)-\hat{y}_{\min}(t+1)}{\hat{y}_{\min}(t+1)} < -\epsilon_{\text{TR}}, \\ J(t-1), & \text{otherwise.} \end{cases}$$

Thus, buy(sell)-signals are triggered upon outbraking the upper(lower) trading corridor affinely constructed based upon the last two local maxima(minima).

Note that for the final trading rules of both TR-Inside and TR-Outside, we only employ the last two local maxima and minima, eventhough we derived $s_{\max}^{(k)}$ in (4.83) (and similarly $t_{\max}^{(k)}$, $s_{\min}^{(k)}$ and $t_{\min}^{(k)}$) for all $k = 1, \ldots, K$. This has the reason that a partition according (4.82) can be constructed either with uniform spacings starting at at time $t - T_{\text{win}}$ and partitioning proceeding forward in time until t, *or*, alternatively, starting at time t and partitioning going backward in time. Interestingly, when testing both methods we found the former method to almost always perform better. We attribute this to the residual δ that typically enlarges the final time-window for $k = K$.

Historical Optimal-based Causal Controller

Given past stock prices historical optimal (HistOpt) trading trajectory can be reconstructed a posteriori with hindsight. This can be done efficiently by graph generation and evaluation. A valid question is whether such optimal trajectories generated up until time $t + 1$ with predicted $\hat{s}(t + 1)$ as final stock price can also be exploited for real-time (RT) stock trading. Thus, trading signals are

$$J(t) = \begin{cases} \tilde{J}(t), & \text{if } \tilde{J}(t-\tau) = \tilde{J}(t),\ \forall \tau = 1, \ldots, T_{\text{HO}} - 1, \\ 0, & \text{otherwise,} \end{cases} \tag{4.84}$$

where $\tilde{J}(t)$ denotes the historical optimal trading trajectory at time t. Tuning parameter T_{HO} determines the number of past consecutive identical trading signals necessary to trigger a buy/sell signal. We refer to this controller as HistOpt-RT.

Table 23: Overfitting illustrated by means of MA-Cross when optimizing parameters on past data. The parameters are $(p_{\text{MA,l}},\ p_{\text{MA,s}})$. They are optimized on the training data (November 28, 2014 until November 26, 2015) and then validated for November 27, 2015 until November 25, 2016. The corresponding return performances in percent are denoted by $f_{\text{train,MA}}$ and $f_{\text{val,MA}}$, respectively. The 30 DAX components are ordered according to increasing absolute performance for the 2-year time period. Buy-and-hold performances $f_{\text{val,B\&H}}$ are given for comparison. The final row indicates average returns (measured in percent).

Stock	Parameters	$f_{\text{train,MA}}$	$f_{\text{val,MA}}$	$f_{\text{val,B\&H}}$
1	(1, 95)	0.0	61.7	7.0
2	(20, 74)	2.2	-29.4	-25.5
3	(9, 38)	14.4	-18.2	-37.3
4	(7, 32)	11.4	-28.1	-39.5
5	(21, 37)	35.4	-19.1	-5.8
6	(18, 11)	15.6	-31.1	-30.4
7	(7, 14)	10.5	-9.3	-7.9
8	(20, 29)	54.8	-20.5	-20.6
9	(9, 18)	24.4	-30.7	-35.4
10	(17, 11)	20.5	-16.5	-24.4
11	(17, 10)	35.9	-30.5	-28.7
12	(14, 26)	13.7	-9.9	4.7
13	(18, 25)	41.0	-4.4	-8.7
14	(6, 26)	29.8	-0.6	-9.8
15	(20, 27)	33.9	-27.2	-16.2
16	(4, 21)	26.8	-30.5	-12.8
17	(17, 27)	27.0	-17.3	-11.0
18	(19, 156)	6.9	12.1	4.9
19	(16, 29)	34.1	1.8	1.7
20	(12, 39)	11.0	1.3	8.3
21	(5, 22)	40.8	7.6	-2.3
22	(12, 20)	12.7	-8.4	2.5
23	(11, 21)	27.9	-24.8	-10.9
24	(4, 9)	29.0	-29.2	-6.0
25	(14, 27)	38.2	1.6	10.5
26	(17, 10)	21.2	-12.2	2.9
27	(9, 23)	32.3	-29.6	7.3
28	(21, 16)	37.5	-9.5	-4.4
29	(17, 43)	46.5	-8.4	13.1
30	(13, 19)	48.8	15.9	50.5
Avg.	–	26.1	-11.4	-7.5

Table 24: Parameter selections for final simulation experiments. See Table 25 for corresponding results.

Controller	Parameters	Values
QP-E+	$(M,\ \alpha)$	$(1,\ 1)$
SMPC-M100	$(M,\ \alpha)$	$(100,\ 10)$
SMPC-DH	M	100
MA-Cross	$(p_{\text{MA},l},\ p_{\text{MA},s})$	$(50,\ 1)$
MA-Sign	$(T_{\text{MA}},\ p_{\text{MA}})$	$(10,\ 100)$
TR-Inside	$(T_{\text{win}},\ p_{\text{TR}},\ \epsilon_{\text{TR}})$	$(261,\ 100,\ 0.01)$
TR-Outside	$(T_{\text{win}},\ p_{\text{TR}},\ \epsilon_{\text{TR}})$	$(261,\ 20,\ 0.03)$
HistOpt-RT	T_{HO}	1

Final Remark

All trading controllers discussed within this section were designed to rely on an estimated step-ahead stock price $\hat{s}(t+1)$. Naturally, price data can arbitrarily be shifted by one sampling interval to the past, thereby making the controllers independent of $\hat{s}(t+1)$ while not prohibiting their applicability. The formulation using $\hat{s}(t+1)$ admits for exploitation of any potential good estimate of step-ahead stock prices. Furthermore, it allows a better comparison with the SMPC-based methods, which are fundamentally based on one-step ahead stock price estimates.

4.4.5 Simulation Experiments

Table 25 summarizes the results for the one-year trading period between November 27, 2015 and November 25, 2016. The average total number of trades per year is denoted by $\bar{N}_{\text{tr}}$. The minimum number of trading days between any two trades is t_{min}. The average, minimum and maximum performance (all measured in $\%$) are $\bar{f}$, f_{min} and f_{max}. The total percentage of positive returns is F_{pos}.

For simulation experiments, we employ the stock prices of the 30 components of the German stock market index DAX between November 28, 2014 and November 25, 2016. For closed-loop trading we only

Table 25: Results for the one-year trading, see Section 4.4.5.

1. Perfect: $\hat{s}(t+1) = s(t+1)$						
Controller	$\bar{N}_{\text{tr}}$	t_{min}	$\bar{f}$	f_{min}	f_{max}	F_{pos}
QP-E+	56	1	150.7	40.8	534.5	100
SMPC-M100	28	1	45.7	0	220.9	100
SMPC-DH	13	1	8.2	0	68.7	100
MA-Cross	21	1	27.7	-3.9	95.0	93.3
MA-Sign	4	26	-8.7	-32.2	50.5	23.3
TR-Inside	2	43	-2.5	-27.2	15.3	60.0
TR-Outside	5	19	7.6	-31.4	55.5	70.0
HistOpt-RT	75	1	133.3	24.1	506.7	100
2. Indifferent: $\hat{s}(t+1) = s(t)$						
Controller	$\bar{N}_{\text{tr}}$	t_{min}	$\bar{f}$	f_{min}	f_{max}	F_{pos}
QP-E+	0	0	0	0	0	100
SMPC-M100	0	0	0	0	0	100
SMPC-DH	2	1	-4	-43.6	2.0	76.7
MA-Cross	1	10	0.7	-19.7	28.9	86.7
MA-Sign	4	26	-8.7	-32.2	50.5	23.3
TR-Inside	2	42	-0.5	-26.5	18.9	60.0
TR-Outside	5	21	-4.2	-42.5	50.8	36.7
HistOpt-RT	75	1	-52.5	-73.8	-30.5	0
3. Random: $\hat{s}(t+1) = s(t) + \eta(t)\frac{1}{t}\sum_{\tau=0}^{t} \lvert s(\tau) - s(\tau-1)\rvert$						
Controller	$\bar{N}_{\text{tr}}$	t_{min}	$\bar{f}$	f_{min}	f_{max}	F_{pos}
QP-E+	57	1	-40.1	-60.7	0.4	3.3
SMPC-M100	20	1	-15.6	-55.3	0.0	26.7
SMPC-DH	25	1	-22.4	-54.8	0.0	23.3
MA-Cross	9	11	-5.0	-36.5	68.6	23.3
MA-Sign	4	26	-8.7	-32.3	50.5	23.3
TR-Inside	2	42	-0.2	-26.4	26.4	56.7
TR-Outside	7	11	-6.7	-39.1	29.7	30.0
HistOpt-RT	116.4	1	-67.2	-80.3	-34.7	0
4. Correct Sign: $\hat{s}(t+1) = s(t) + 10\xi(t)\text{sign}(s(t+1) - s(t))$						
Controller	$\bar{N}_{\text{tr}}$	t_{min}	$\bar{f}$	f_{min}	f_{max}	F_{pos}
QP-E+	117	1	74.7	-25.9	400.4	86.7
SMPC-M100	119	1	57.8	-29.6	338.6	90.0
SMPC-DH	65	1	3	-34.1	74.5	60.0
MA-Cross	21	1	22.0	-17.1	171.8	73.3
MA-Sign	4	26	-8.7	-32.3	50.5	23.3
TR-Inside	6	21	-10.9	-32.2	16.2	23.3
TR-Outside	51	1	28.3	-32.7	188.1	70.0
HistOpt-RT	125	1	69.3	-28.9	380.7	83.3
5. Wrong Sign: $\hat{s}(t+1) = s(t) - 10\xi(t)\text{sign}(s(t+1) - s(t))$						
Controller	$\bar{N}_{\text{tr}}$	t_{min}	$\bar{f}$	f_{min}	f_{max}	F_{pos}
QP-E+	117.4	1	-93.5	-98.7	-87.6	0
SMPC-M100	119	1	-93.4	-98.6	-88.5	0
SMPC-DH	120	1	-93.7	-98.7	-88.8	0
MA-Cross	11	3	-16.9	-40.8	2.7	3.3
MA-Sign	4	26	-8.7	-32.3	50.5	23.3
TR-Inside	5	28	-3.1	-28.7	26.1	40.0
TR-Outside	56	1	-64.3	-97.1	-19.3	0
HistOpt-RT	136	1	-94.9	-98.7	-91.4	0
Global Optimum (Trading w/ Hindsight)/Buy-and-Hold						
Controller	$\bar{N}_{\text{tr}}$	t_{min}	$\bar{f}$	f_{min}	f_{max}	F_{pos}
HistOpt	42	1	192.6	64.7	609.5	100
Buy-and-Hold	1	0	-7.5	-39.5	50.5	36.7

considered the past year and initialized $t = 0$ on November 27, 2015. Nevertheless, previous price data was still relevant for the generation of measures such as moving averages at $t = 0$. All data was drawn from `finance.yahoo.com`, see Figure 95 for visualization.

Closed-loop Trading Results

Each stock is traded separately and the portfolio with transition dynamics according to Section 4.4.2 is initialized with $Z(0) = \begin{bmatrix} 0 & M_0 & 0 & M_0 \end{bmatrix}$ where $M_0 = 100000$€. We compare eight different controllers and five different methods that we use for the prediction of $\hat{s}(t+1)$. In addition we state the results for a *buy-and-hold* strategy (investing maximally into the stock at $t = 0$ and consequently holding the investment throughout), and for the global optimal trading result (trading with hindsight). We assume proportional transaction costs $\epsilon = 0.01$ identical for both buying and selling of stocks. Performance is defined by $f = \frac{W(T)-M_0}{M_0} 100$ with T the final trading date.

Regarding parameter selections for the genetic algorithms, we tested three settings. First, we optimized parameters on training data (November 28, 2014 until November 26, 2015) and then validated for November 27, 2015 until November 25, 2016. Thus, for each stock and for each controller individual parameters were selected. Second, we recursively updated parameters. Thus, every 100 days (we also tested 20 and 50 days) we recomputed parameters optimized on past data of one year at that time. Third, we arbitrarily chose a fixed parameter set for each controller and used this for the trading of *all* 30 stocks. For both the first and the second approach strong overfitting could be observed, see Table 23. We therefore opted for fixed parameter selections for the trading of all stocks. More conservative parameter selections, such as, e.g., larger MA-windows $p_{\text{MA,l}}$, performend on average better. For HistOpt-RT, we intentionally chose $T_{\text{HO}} = 1$, which is the most aggressive but least robust choice as outlined in the following. The parameter selections employed for final simulation experiments are summarized in Table 24.

Closed-loop trading results are summarized in Table 25. Because of the importance of step-ahead predictions and for robustness considera-

tions, we compare five different versions for $\hat{s}(t+1)$. Ideally (but unrealistically in general), $s(t+1)$ is estimated perfectly, i.e., $\hat{s}(t+1) = s(t+1)$. This case is noncausal. Nevertheless, it serves as an important benchmark. Case 4 and 5 ("correct" and "wrong sign prediction") are likewise noncausal since $s(t+1)$ is not known at time t. Guiding notion for their introduction was to analyze influence of correct trend prediction (up or down) for the price one time-step ahead but without knowledge of exact level of price rise or fall. We therefore add multiplicative time-varying perturbation $10\xi(t)$ with $\xi(t) \sim \mathcal{U}(0,1)$ uniformly distributed. Case 2 ("indifferent") uses the current stock price as the estimate for the ext time-step ahead. Case 3 ("random") randomly perturbs $s(t)$ as the estimate for $\hat{s}(t+1)$, whereby $\eta(t) \sim \mathcal{N}(0,1)$ normally distributed. Results are discussed in the next section. Importantly, we remark that only 36.7% of the 30 DAX components rose, i.e., $s(T) > s(0)$, for the one-year trading period considered between November 27, 2015 and November 25, 2016.

Discussion

Several observations can be made from Table 25. Let us first discuss the results for SMPC-based stock trading and HistOpt-RT. For the ideal case of perfect $s(t+1)$ knowledge, excellent results can be obtained. Both for QP-E+ and HistOpt-RT. Performances are even in range with the global optimum (HistOpt) despite only *one-step* ahead price knowledge. By reduction of α from 10 to 1, quasi identical performance to QP-E+ is achieved by SMPC-M100, i.e., $(\bar{f}, f_{\min}, f_{\max}) = (139.4, 42.7, 533.7)$. We selected $\alpha = 10$ to illustrate its role in adding robustness. This becomes apparent for the random $\hat{s}(t+1)$ prediction method (case 3): while $\bar{f} = -40.1\%$ for QP-E+, it is $\bar{f} = -15.6\%$ for SMPC-M100, and additionally yielding 26.7% positive returns overall. For the indifferent prediction $\hat{s}(t+1) = s(t)$, QP-E+ and SMPC-M100 *never* enter a trade. This was expected. Both optimization problem formulations essentially rely on the mean difference between $s(t)$ and $\hat{s}(t+1)$. Characteristically they do not consider *any* past data points except the current price $s(t)$. This is in contrast to HistOpt-RT where all available past data up until t is searched for the optimal trading trajectory. SMPC-DH was found

to not be competitive with the other two SMPC-based controllers (QP-E+ and SMPC-M100), neither with respect to performance nor robustness. Nevertheless, its framework is favorable in that more (and better) heuristics can easily be incorporated by adjusting reference scenarios $p^j(t+1),\ j=1,\ldots,M$.

Of great relevance to SMPC-based trading strategies and HistOpt-RT are the experiments for correct and wrong sign predictions (case 4 and 5). Importantly, they indicate that perfect one-step ahead *sign* predictions of price changes $s(t+1)-s(t)$ are sufficient for excellent results. Thus, the precise level of increase or decrease in stock prices is not necessarily required. For illustration, consider the average gain of 74.7% *per stock* for QP-E+ despite the fact that only 36.7% of all 30 DAX components actually rose during the past year and moreover on average yielding -7.5% (see the Buy-and-Hold strategy). Even more important are the results for case 5, i.e., when at every trading instant wrongly predicting the direction of change in stock prices. For *all* SMPC-based trading methods and *HistOpt-RT*, *after just one year*, *at least* 93.5% of all initial wealth is lost.

Finally, note that for the SMPC-based methods and HistOpt-RT the minimum time-span between any two trades is always 1 day (except for case 2 when there are no trades at all). Furthermore, the average number of trades per year, $\bar{N}_{\text{tr}}$, is considerably larger in comparison to the genetic algorithms. Throughout experiments, more robust performance could be observed for the genetic algorithms. For the given parameter choices, MA-Cross appeared to be best suited to exploit potential knowledge of future stock prices (case 1). Encouraging are the returns for MA-Cross and TR-Inside for the causal prediction case, i.e., $\hat{s}(t+1)=s(t)$. Despite the fact that only 36.7% of all 30 DAX components rose, the two controllers yielded positive returns for 86.7% and 60%, respectively. For random price-ahead predictions (case 3), the performance of all four MA- and TR-based controllers was comparable to the Buy-and-Hold method, with TR-Inside best on average and with respect to worst-case losses.

Before concluding, let us remark some realistic success ratios reported in the literature for correct sign predictions of step-ahead price difference $s(t+1)-s(t)$. In [217], support vector machines (SVM) in combination

with 12 technical indicators (such as Williams %R, stochastic %K, disparity, etc.) are used to predict the direction of change in the daily Korea composite stock price index (KOSPI). For validation data and their best tuning parameter choices, they report a prediction performance between 50.1% and 57.8%. The same author mentioned similar results in earlier work [218].

4.4.6 Conclusion

For stock trading, the general class of genetic algorithms appears more suitable than methods based on stochastic model predictive control. The former class is signficantly more robust. A SMPC-approach is justifiable only for consistently perfect prediction of direction of price changes. The relations and differences between using SMPC for dynamic hedging and stock trading were discussed.

Findings motivate the following:

1. A detailed analysis of scenarios when MA- and TR-based algorithms fail and succeed, respectively.

2. An artificial and automated generation of genetic trading algorithms to further improve performance and robustness [6].

3. The usage of options for their predictable worst-case loss [150].

Subject of future research may also be the application of genetic algorithms to both multi-asset portfolio optimization and dynamic option hedging.

4.4.7 Hierarchical Controller Parametrization

There are two hierarchical layers. These are:

1. A method to predict the step-ahead stock price;

2. Any of the eight trading methods from Sections 4.4.3 and 4.4.4.

4.5 Discussion of Chapter

Summary

Findings can be summarized as follows. First, SMPC is a good strategy for both dynamic option hedging and single-asset trading when good one-step-ahead stock price estimates are available. Consisently correct step-ahead *sign* predictions are already sufficient to generate excellent results. Second, however, for consistenly wrong step-ahead *sign* predictions, catastrophic returns result. Thus, discussed SMPC methods are heavily relying on the step-ahead prediction quality. Third, for dynamic option hedging and in combination with SMPC, the simplest scenario generation scheme, in which the current stock price is treated as the step-ahead mean estimate, performed best. This simple scheme outperformed both extrapolating a SVR- and a lognormal-fitted model, which both did not well enough predict step-ahead stock prices. Fourth, for single-asset stock trading and among the genetic algorithms tested, the "moving average crossing" algorithm performed best. Nevertheless, it is prone to perform poorly when its parameters are tuned excessively on training data. In fact, overfitting on training data is *the* problem for all parameterized methods. These included the genetic algorithms, but also the SVR- and the lognormal-based stock price prediction schemes. Therefore, for robustness, a larger moving average horizon of, for example, 50 days is recommended. Fifth, optimization of trading trajectories with hindsight indicated the enormous potential of knowledgable financial decision taking. Dependent on transaction cost levels, for optimal performance and under the assumption of perfect knowledge, short-term trends must be exploited with active and frequent trading, typically upon single- or low double-digit percentage gains for low or higher transaction costs, respectively. Finally, a method was developed to automatically and cost-efficiently combine a set of available vanilla options from a database to approximate a desired profit profile. This method is beneficial for two reasons. First, the combination step is deterministic. Second, the worst-case loss and the scenario (i.e., the underlying stock price) in which this worst-case is achieved is also deterministic.

Future Work

There are three main avenues for future work. First, quantitative finance is centered critically around the ability to *anticipate*. This cannot be circumvented and holds for both dynamic option hedging and trading for profit. Thus, in a first step, improved prediction methods must be further analyzed. Here, possible success ratio of between 50% and at most 60% appear to be a limit. In [217], a correct prediction of the direction of change in the daily Korea composite stock price index (KOSPI) ranged between 50.1% to 57.8% for validation data and their best tuning parameters. For future work, reinforcement learning (RL) may be leveraged. First, a parametrization is needed. A good parametrization may be based on RNNs that enable temporal data processing. Second, input vectors must be defined. These may, for example, include sequences of past prices and trading volumes. Third, training for the identification of parameters can then be conducted by a RL method. For example, by TSHC from [142]. Finally, one important aspect to address is the time frame. It likely may be that predictions are easier to make for longer time frames (less noise), rather than predictions for the day-ahead price.

The second topic is *underlying-screening*. All derivative contracts are by definition dependent on an *underlying* (for example, a stock price). Therefore, suitable underlyings must first be identified ("stock picking"). Here, one approach is *functional clustering*. This also is related to *pairs trading*. Another approach is to again make use of reinforcement learning. Softmax classifiers in the output layer may indicate investment grades of various stocks, whereby *one* neural network is comprehensively modeling a regional economy or sector class. Note that suitable stocks are these which are best to *anticipate* according to above first remark.

Finally, in a hierarchical optimization scheme, the last layer involves the *realization* of trading decisions. Here, methods based on SMPC and derivative combination seem to be appropriate. SMPC is suitable for its ability to incorporate various constraints. Derivative combinations are suitable for their predictable worst-case scenarios (*pessimization* approach). In the latter perspective, also *future contracts* will be revised.

Chapter 5

Conclusion

Summary

Hierarchical planning and stochastic optimization algorithms with applications to self-driving vehicles and finance were discussed. A diverse set of mathematical tools was employed. Special focus was on model predictive control in both the deterministic and stochastic setting.

Unification of the different topics motion planning, vehicle routing and quantitative finance was achieved by a *layered optimization approach*. Methods differ in their parametrization of each layer, input (data) and output (decision variables) to each layer, and sampling time (update frequency). Different layer parametrizations discussed include QPs, LPs, SLPs, stochastic MPC, Kalman Filters, SVRs, neural networks, BILPs, IPs and various VRPs.

The two most natural hierarchical layers are *high-level planning* and *lower-level execution*. For example, for self-driving vehicles this implies high-level route planning and lower-level vehicle motion planning for realization of the path plan.

In-depth conclusions on single- and multi-vehicle motion planning, vehicle routing and quantitative finance were provided in Sections 2.4, 3.6 and 4.5, respectively.

Future Work

Next work would include the further exploration of a) *neural networks* for the design of continuous controllers using reinforcement learning, and b) *optimization via simulation* for the solution of large-scale integer programs for logistics. Note that the presented TSHC-algorithm can be used as a framework for both topics. The difference lies in the solution parametrization. For logistical applications, integer programs can be used as parametrization. For control applications, neural networks can be used as parametrization to enable continuous control. Attractive features of this approach are scalability, the theoretically unlimited function approximation capability of neural networks, simplicity, and further developing hardware opportunities for large-scale simulations. For continuous control, the comparison between, on one hand, model-based offline encoding of motion primitives in neural networks using high-fidelity system models, and, on the other hand, model-based online optimization using simplified system models may be one main topic of research.

Finally, c) the importance of the planning step with *setpoint* selections is emphasized. It represents the first of the hierarchical optimization layers. Within the automated vehicles context, it corresponds to the setpoint selection as a function of filtered extero- and proprioceptive measurements. Importantly, this setpoint must always be selected in *real-time*. In general, for the planning step, *graph-based* search methods (such as Algorithms 1 and 2 in the automotive setting) seem to be appropriate. Graph-based planning is fast (greedy search), can handle non-convex problems, and permits to derive worst-case search complexities when limiting the maximal graph-size for real-time operation. In the automated vehicles context, setpoints may be selected as a function of camera and lidar measurements, for example, as the center of permissible driving area. In the financial setting, setpoints correspond to the selection of desired reference profit profiles. Once selected, these can be realized through the combinations of derivative contracts. To summarize, part of future work would focus on the mapping from various sources of measurements to setpoints. This applies for both the self-driving and the financial domain.

References

[1] T. Achterberg, "SCIP: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.

[2] O. Ahumada and J. R. Villalobos, "Application of planning models in the agri-food supply chain: A review," *European Journal of Operational Research*, vol. 196, no. 1, pp. 1–20, 2009.

[3] R. M. Aiex, M. G. C. Resende, and C. C. Ribeiro, "Probability distribution of solution time in grasp: an experimental investigation," 2003. [Online]. Available: www.MirraMarx.com/graspintime.html

[4] T. Akiba, S. Suzuki, and K. Fukuda, "Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes," *arXiv preprint arXiv:1711.04325*, 2017.

[5] A. Al Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning," in *IEEE ITSC*, 2010, pp. 306–311.

[6] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Financial Economics*, vol. 51, no. 2, pp. 245–271, 1999.

[7] A. Altarovici, J. Muhle-Karbe, and H. M. Soner, "Asymptotics for fixed transaction costs," *Finance and Stochastics*, vol. 19, no. 2, pp. 363–414, 2015.

[8] F. Altché, P. Polack, and A. de La Fortelle, "High-speed trajectory planning for autonomous vehicles using a simple dynamic model," *arXiv: 1704.01003*, 2017.

[9] B. D. Anderson and J. B. Moore, "Optimal filtering," *Englewood Cliffs, USA: Prentice Hall*, vol. 21, pp. 22–95, 1979.

[10] C. W. Anderson, "Learning to control an inverted pendulum using neural networks," *IEEE Control Systems Magazine*, vol. 9, no. 3, pp. 31–37, 1989.

[11] D. L. Antille, J. M. Bennett, and T. A. Jensen, "Soil compaction and controlled traffic considerations in australian cotton-farming systems," *Crop and Pasture Science*, vol. 67, no. 1, pp. 1–28, 2016.

[12] D. Antille, D. Ansorge, M. Dresser, and R. Godwin, "Soil displacement and soil bulk density changes as affected by tire size," *Transactions of the ASABE*, vol. 56, no. 5, pp. 1683–1693, 2013.

[13] B. Asadi and A. Vahidi, "Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 707–714, 2011.

[14] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *IEEE/RSJ Conference on Intelligent Robots and Systems*, 2012, pp. 1917–1922.

[15] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[16] J. Backman, T. Oksanen, and A. Visala, "Navigation system for agricultural machines: nonlinear model predictive path tracking," *Computers and Electronics in Agriculture*, vol. 82, pp. 32–43, 2012.

[17] ——, "Path generation method with steering rate constraint," in *Proc. International Conference on Precision Agriculture (ICPA)*, USA: Indianapolis, 2012.

[18] ——, "Applicability of the ISO 11783 network in a distributed combined guidance system for agricultural machines," *Biosystems Engineering*, vol. 114, no. 3, pp. 306–317, 2013.

[19] J. Backman, P. Piirainen, and T. Oksanen, "Smooth turning path generation for agricultural vehicles in headlands," *Biosystems Engineering*, vol. 139, pp. 76–86, 2015.

[20] V. L. Bageshwar, W. L. Garrard, and R. Rajamani, "Model predictive control of transitional maneuvers for adaptive cruise control vehicles," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 5, pp. 1573–1585, 2004.

[21] O. C. Barawid, A. Mizushima, K. Ishii, and N. Noguchi, "Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application," *Biosystems Engineering*, vol. 96, no. 2, pp. 139–149, 2007.

[22] B. R. Barmish, "On stock market modeling and trading: New problems for the control field," in *Proc. 47th IEEE Conf. on Decision and Control*, 2008, pp. 13–14, CDC semi-plenary lecture.

[23] ——, "On trading of equities: A robust control paradigm," in *Proc. 17th IFAC World Congress*, 2008, pp. 1621–1626.

[24] B. R. Barmish, S. Condie, D. Materassi, J. A. Primbs, and S. Warnick, "On nasdaq order book dynamics: New problems for the control field," in *IEEE American Control Conference*. American Automatic Control Council (AACC), 2016, pp. 5671–5672.

[25] B. R. Barmish and J. A. Primbs, "Stock trading via feedback control," *Encyclopedia of Systems and Control*, pp. 1357–1364, 2015.

[26] L. D. Baskar, B. De Schutter, J. Hellendoorn, and Z. Papp, "Traffic control and intelligent vehicle highway systems: a survey," *Intelligent Transport Systems*, vol. 5, no. 1, pp. 38–52, 2011.

[27] C. Basnet, L. R. Foulds, and J. Wilson, "Scheduling contractors' farm-to-farm crop harvesting operations," *International Transactions in Operational Research*, vol. 13, no. 1, pp. 1–15, 2006.

[28] M. T. Batte and M. R. Ehsani, "The economics of precision guidance with auto-boom control for farmer-owned agricultural sprayers," *Computers and Electronics in Agriculture*, vol. 53, no. 1, pp. 28–44, 2006.

[29] J. E. Beasley, N. Meade, and T.-J. Chang, "An evolutionary heuristic for the index tracking problem," *European Journal of Operational Research*, vol. 148, no. 3, pp. 621–643, 2003.

[30] B. Becker and H. Giese, "On safe service-oriented real-time coordination for autonomous vehicles," in *IEEE Symposium on Object Oriented Real-Time Distributed Computing*, 2008, pp. 203–210.

[31] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[32] A. Bemporad, "Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form," *IEEE Trans. Automatic Control*, vol. 49, no. 5, pp. 832–838, 2004.

[33] ——, "Model-based predictive control design: New trends and tools," in *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, CA, 2006, pp. 6678–6683.

[34] A. Bemporad, L. Bellucci, and T. Gabbriellini, "Dynamic option hedging via stochastic model predictive control based on scenario simulation," *Quantitative Finance*, vol. 14, no. 10, pp. 1739–1751, 2014.

[35] A. Bemporad, T. Gabbriellini, L. Puglia, and L. Bellucci, "Scenario-based stochastic model predictive control for dynamic option hedging," in *Proc. 49th IEEE Conf. on Decision and Control*, Atlanta, GA, USA, 2010, pp. 6089–6094.

[36] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[37] ——, "Robust model predictive control: A survey," in *Robustness in Identification and Control*, ser. Lecture Notes in Control and Information Sciences, A. Garulli, A. Tesi, and A. Vicino, Eds. Springer-Verlag, 1999, no. 245, pp. 207–226.

[38] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[39] A. Bemporad, M. Morari, and N. Ricker, *Model Predictive Control Toolbox for Matlab – User's Guide*. The Mathworks, Inc., 2004, http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

[40] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[41] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[42] A. Bemporad, L. Puglia, and T. Gabbriellini, "A stochastic model predictive control approach to dynamic option hedging with transaction costs," in *IEEE American Control Conference*, San Francisco, CA, USA, 2011, pp. 3862–3867.

[43] A. Bemporad and C. Rocchi, "Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles," in *Proceedings of the 18th IFAC World Congress*, 2011, pp. 11 900–11 906.

[44] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *International Conference on Machine Learning*. ACM, 2009, pp. 41–48.

[45] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *IEEE Conference on Robotics and Automation*, vol. 1, 2001, pp. 271–276.

[46] ——, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and autonomous systems*, vol. 41, no. 2, pp. 89–99, 2002.

[47] D. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[48] D. Bertsimas, L. Kogan, and A. Lo, "Hedging derivative securities and incomplete markets: an ϵ-arbitrage approach," *Operations Research*, vol. 49, no. 3, pp. 372–397, 2001.

[49] F. Black and M. Scholes, "Pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973.

[50] ——, "The pricing of options and corporate liabilities," *The Journal of Political Economy*, pp. 637–654, 1973.

[51] B. Blackmore, S. Fountas, T. Gemtos, and H. Griepentrog, "A specification for an autonomous crop production mechanization system," in *International Symposium on Application of Precision Agriculture for Fruits and Vegetables*, USA: Orlando, 2008, pp. 201–216.

[52] BMW, "Traffic jam assistant," http://www.bmw.com /com/en/newvehicles/x/x5/2013/\showroom/driver_ assistance/traffic_jam_assistant.html, 2013.

[53] D. Bochtis, "Machinery management in bio-production systems: planning and scheduling aspects," *Agricultural Engineering International: CIGR Journal*, vol. 12, no. 2, pp. 55–63, 2010.

[54] ——, "Satellite based technologies as key enablers for sustainable ict-based agricultural production systems," *Procedia Technology*, vol. 8, pp. 4–8, 2013.

[55] D. Bochtis, H. Griepentrog, S. Vougioukas, P. Busato, R. Berruto, and K. Zhou, "Route planning for orchard operations," *Computers and Electronics in Agriculture*, vol. 113, pp. 51–60, 2015.

[56] D. Bochtis and C. Sørensen, "The vehicle routing problem in field logistics part i," *Biosystems Engineering*, vol. 104, no. 4, pp. 447–457, 2009.

[57] ——, "The vehicle routing problem in field logistics: Part ii," *Biosystems Engineering*, vol. 105, no. 2, pp. 180–188, 2010.

[58] D. Bochtis, C. Sørensen, and O. Green, "A DSS for planning of soil-sensitive field operations," *Decision Support Systems*, vol. 53, no. 1, pp. 66–75, 2012.

[59] D. Bochtis, C. Sørensen, O. Green, D. Moshou, and J. Olesen, "Effect of controlled traffic on field efficiency," *Biosystems Engineering*, vol. 106, no. 1, pp. 14–25, 2010.

[60] D. Bochtis, C. Sørensen, and S. Vougioukas, "Path planning for in-field navigation-aiding of service units," *Computers and Electronics in Agriculture*, vol. 74, no. 1, pp. 80–90, 2010.

[61] D. Bochtis, C. G. Sørensen, P. Busato, and R. Berruto, "Benefits from optimal route planning based on b-patterns," *Biosystems Engineering*, vol. 115, no. 4, pp. 389–395, 2013.

[62] D. Bochtis and S. Vougioukas, "Minimising the non-working distance travelled by machines operating in a headland field pattern," *Biosystems Engineering*, vol. 101, no. 1, pp. 1–12, 2008.

[63] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[64] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*. Macmillan London, 1976, vol. 290.

[65] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[66] M. Bouroche, B. Hughes, and V. Cahill, "Real-time coordination of autonomous vehicles," in *IEEE Conference on Intelligent Transportation Systems*, 2006, pp. 1232–1239.

[67] V. Boyarshinov and M. Magdon-Ismail, "Efficient computation of optimal trading strategies," *arXiv preprint arXiv:1009.4683*, 2010.

[68] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[69] P. Boyle, "Options: A Monte Carlo approach," *Journal of Financial Economics*, vol. 4, no. 3, pp. 323–338, 1977.

[70] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles," *CEP*, vol. 61, pp. 307–316, 2017.

[71] J. C. Butcher, *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[72] G. C. Calafiore, "Multi-period portfolio optimization with linear control policies," *Automatica*, vol. 44, pp. 2463–2473, 2008.

[73] P. Carr, *Dynamic and Static Hedging of Exotic Equity Options*, 1999. [Online]. Available: http://www.math.nyu.edu/research/carrp/papers

[74] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 2335–2340.

[75] A. Carvalho, S. Lefèvre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty-a control perspective," *European Journal of Control*, 2015.

[76] D. M. Chance and R. Brooks, *Introduction to derivatives and risk management*. Cengage Learning, 2015.

[77] J. S. Chaput and L. H. Ederington, "Option spread and combination trading," *Available at SSRN 296036*, 2002.

[78] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management science*, vol. 6, no. 1, pp. 73–79, 1959.

[79] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[80] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain inspired cognitive model with attention for self-driving cars," *arXiv preprint arXiv:1702.05596*, 2017.

[81] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[82] J. Clossey, G. Laporte, P. Soriano *et al.*, "Solving arc routing problems with turn penalties," *Journal of the Operational Research Society*, vol. 52, no. 4, pp. 433–439, 2001.

[83] J. Conesa-Muñoz, J. M. Bengochea-Guevara, D. Andujar, and A. Ribeiro, "Route planning for agricultural tasks: A general approach for fleets of autonomous vehicles in site-specific herbicide applications," *Computers and Electronics in Agriculture*, vol. 127, pp. 204–220, 2016.

[84] J. Conesa-Muñoz, G. Pajares, and A. Ribeiro, "Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment," *Expert Systems with Applications*, vol. 54, pp. 364–378, 2016.

[85] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press Cambridge, 2001, vol. 6.

[86] G. Cornuejols and R. Tutuncu, *Optimization Methods in Finance*. New York, USA: Cambridge University Press, 2007.

[87] J. C. Cox, J. E. Ingersoll, and S. A. Ross, "A theory of the term structure of interest rates," *Econometrica: Journal of the Econometric Society*, pp. 385–407, 1985.

[88] J. C. Cox and M. Rubinstein, *Options markets*. Prentice Hall, 1985.

[89] A. Dain-Owens, M. Kibblewhite, M. Hann, and R. Godwin, "The risk of harm to archaeological artefacts in soil from dynamic subsurface pressures generated by agricultural operations: Experimental studies," *Archaeometry*, vol. 55, no. 6, pp. 1175–1186, 2013.

[90] W. Day, "Engineering advances for input reduction and systems management to meet the challenges of global food and farming futures," *The Journal of Agricultural Science*, vol. 149, no. S1, pp. 55–61, 2011.

[91] S. Di Cairano, U. Kalabić, and K. Berntorp, "Vehicle tracking control on piecewise-clothoidal trajectories by MPC with guaranteed error bounds," in *IEEE CDC*, 2016, pp. 709–714.

[92] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[93] M. D. Dikaiakos, A. Florides, T. Nadeem, and L. Iftode, "Location-aware services over vehicular ad-hoc networks using car-to-car communication," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 8, pp. 1590–1602, 2007.

[94] Q. T. Dinh and M. Diehl, "An application of sequential convex programming to time optimal trajectory planning for a car motion," in *IEEE CDC*, December 2009, pp. 16–18.

[95] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *IJRR*, vol. 29, no. 5, pp. 485–501, 2010.

[96] V. V. Dombrovskii, D. V. Dombrovskii, and E. A. Lyashenko, "Predictive control of random-parameter systems with multiplicative noise. Application to investment portfolio optimization," *Automation and Remote Control*, vol. 66, no. 4, pp. 583–595, 2005.

[97] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[98] D. Duffie, *Dynamic asset pricing theory*. Princeton University Press Princeton, NJ, 1996.

[99] C. Edirisinghe, V. Naik, and R. Uppal, "Optimal replication of options with transactions costs and trading restrictions," *Journal of Financial and Quantitative Analysis*, vol. 28, no. 1, pp. 117–138, 1993.

[100] D. M. F. Edwards, P. A. Madden, and I. R. McDonald, "Parallel grasp with path-relinking for job shop scheduling," *Mol. Phys.*, vol. 51, no. 5, pp. 1141–1151, 1984.

[101] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part i: The chinese postman problem," *Operations Research*, vol. 43, no. 2, pp. 231–242, 1995.

[102] ——, "Arc routing problems, part ii: The rural postman problem," *Operations research*, vol. 43, no. 3, pp. 399–414, 1995.

[103] R. Fahlenbrach and P. Sandås, "Does information drive trading in option strategies?" *Journal of Banking & Finance*, vol. 34, no. 10, pp. 2370–2385, 2010.

[104] P. Falcone, F. Borrelli, H. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 862–875, 2008.

[105] P. Falcone, M. Ali, and J. Sjöberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1352–1361, 2011.

[106] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on control systems technology*, vol. 15, no. 3, pp. 566–580, 2007.

[107] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *IEEE CDC*, 2007, pp. 2980–2985.

[108] E. F. Fama, "Market efficiency, long-term returns, and behavioral finance," *Journal of Financial Economics*, vol. 49, no. 3, pp. 283–306, 1998.

[109] S. Fedotov, "Stochastic optimization approach to options pricing," in *Proc. American Contr. Conf.*, Chicago, IL, Jun. 1999, pp. 1450–1453.

[110] S. Fedotov and S. Mikhailov, "Option pricing for incomplete markets via stochastic optimization: transaction costs, adaptive control and forecast," *International Journal of Theoretical and Applied Finance*, vol. 4, no. 1, pp. 179–195, 1999.

[111] J.-C. Ferrer, A. Mac Cawley, S. Maturana, S. Toloza, and J. Vera, "An optimization approach for scheduling wine grape harvest operations," *International Journal of Production Economics*, vol. 112, no. 2, pp. 985–999, 2008.

[112] C. Filippi, R. Mansini, and E. Stevanato, "Mixed integer linear programming models for optimal crop selection," *Computers & Operations Research*, vol. 81, pp. 26–39, 2017.

[113] Financial Times, "Wheat price falls to lowest level in a decade," https://www.ft.com/content/75fdb856-6b0c-11e6-ae5b-a7cc5dd5a28c, Aug. 2016.

[114] M. Fliess and C. Join, "A mathematical proof of the existence of trends in financial time series," *Systems Theory: Modelling, Analysis and Control*, pp. 43–62, 2009.

[115] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.

[116] R. Franke, "Omuses, a tool for the optimization of multistage systems and hqp, a solver for sparse nonlinear optimization," 1998, technischer Bericht. Institut für Automatisierungs-und Systemtechnik, Technische Universität Ilmenau, Deutschland.

[117] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *IEEE ECC*, 2013, pp. 4136–4141.

[118] E. Frazzoli, M. A. Dahleh, and E. Feron, "A hybrid control architecture for aggressive maneuvering of autonomous helicopters," in *IEEE Conference on Decision and Control*, vol. 3, 1999, pp. 2471–2476.

[119] M. C. Fu, F. W. Glover, and J. April, "Simulation optimization: a review, new developments, and applications," in *IEEE Winter Simulation Conference*. IEEE, 2005, pp. 13–pp.

[120] J. Funke and J. C. Gerdes, "Simple clothoid lane change trajectories for automated vehicles incorporating friction constraints," *Jrnl. of Dyn. Sys., Meas. and Ctrl.*, vol. 138, no. 2, pp. 021 002–021 002–9, 2016.

[121] ——, "Simple clothoid lane change trajectories for automated vehicles incorporating friction constraints," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 2, p. 021002, 2016.

[122] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler, and B. Huhnke, "Up to the limits: Autonomous audi tts," in *IEEE Intelligent Vehicles Symposium*, 2012, pp. 541–547.

[123] A. A. Gaivoronski, S. Krylov, and N. Van der Wijst, "Optimal portfolio selection and dynamic benchmark tracking," *European Journal of Operational Research*, vol. 163, no. 1, pp. 115–131, 2005.

[124] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012.

[125] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *ASME Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2010, pp. 265–272.

[126] H. Geering, G. Dondi, F. Herzog, and S. Keel, "Stochastic systems," *Course script*, 2011.

[127] M. Gendreau, A. Hertz, and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Management Science*, vol. 40, no. 10, pp. 1276–1290, 1994.

[128] M. Gerla and L. Kleinrock, "Vehicular networks and the future of the mobile internet," *Computer Networks*, vol. 55, no. 2, pp. 457–469, 2011.

[129] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of Machine Learning Research*, vol. 3, no. Aug, pp. 115–143, 2002.

[130] T. V. Gestel, J. Suykens, D. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. D. Moor, and J. Vandewalle, "Financial time series prediction using least squares support vector machines within the evidence framework," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 809–821, 2001.

[131] T. D. Gillespie, "Vehicle dynamics," *Warren dale*, 1997.

[132] T. Glasmachers, "Limits of end-to-end learning," *arXiv preprint arXiv:1704.08305*, 2017.

[133] J. J. Glen, "Mean-variance portfolio rebalancing with transaction costs and funding changes," *Journal of the Operational Research Society*, vol. 62, pp. 667–676, 2011.

[134] F. Glover and C. C. Ribeiro, "Multi-start and strategic oscillation methods – principles to exploit adaptive memory," in *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, 2nd ed., M. Laguna and J. L. Gonzáles-Velarde, Eds. Boston, MA: Kluwer Academic, 2000, pp. 1–24.

[135] GM: Cadillac, "Introducing super cruise," http://www.cadillac.com/sedans/ct6-sedan.html, 2017.

[136] L. Gomes, "When will google's self-driving car really be ready? it depends on where you live and what you mean by" ready"[news]," *IEEE Spectrum*, vol. 53, no. 5, pp. 13–14, 2016.

[137] R. E. Gomory and W. J. Baumol, "Integer programming and pricing," *Econometrica: Journal of the Econometric Society*, pp. 521–550, 1960.

[138] J. Gondzio, R. Kouwenbergb, and T. Vorst, "Hedging options under transaction costs and stochastic volatility," *Journal of Economic Dynamics & Control*, vol. 27, pp. 1045–1068, 2003.

[139] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 17, no. 4, pp. 1135–1145, 2016.

[140] P. Gonzalez-de Santos, A. Ribeiro, C. Fernandez-Quintanilla, F. Lopez-Granados, M. Brandstoetter, S. Tomic, S. Pedrazzi, A. Peruzzi, G. Pajares, G. Kaplanis *et al.*, "Fleets of robots for environmentally-safe pest control in agriculture," *Precision Agriculture*, pp. 1–41, 2016.

[141] J. Gozálvez, M. Sepulcre, and R. Bauza, "IEEE 802.11p vehicle to infrastructure communications in urban environments," *IEEE Commun. Mag.*, vol. 50, no. 5, 2012.

[142] M. Graf Plessen, "Automating vehicles by deep reinforcement learning using task separation with hill climbing," *arXiv preprint arXiv:1711.10785*, 2017, (Submitted).

[143] ——, "Coordination of harvesting and transport units for area coverage," Jun. 4 2017, pCT/IB2017/051899.

[144] ——, "Coupling of crop assignment and vehicle routing for harvest planning in agriculture," *arXiv preprint arXiv:1703.08999*, 2017, (Submitted).

[145] ——, "Partial field coverage based on two path planning patterns," *arXiv preprint arXiv:1707.07649*, 2017, (Draft).

[146] ——, "Path planning for area coverage," Jun. 8 2017, wO Patent App. PCT/EP2016/072,966. [Online]. Available: https://encrypted.google.com/patents/WO2017092904A1?cl=en

[147] ——, "System and method for navigation guidance of a vehicle in an agricultural field," Jun. 8 2017, wO Patent App. PCT/EP2016/072,968. [Online]. Available: https://www.google.com/patents/WO2017092905A1?cl=en

[148] ——, "Trajectory planning of automated vehicles in tube-like road segments," in *IEEE Conference on Intelligent Transportation Systems*, 2017, pp. 83–88.

[149] M. Graf Plessen and A. Bemporad, "Shortest path computations under trajectory constraints for ground vehicles within agricultural fields," in *IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 1733–1738.

[150] ——, "Parallel investments in multiple call and put options for the tracking of desired profit profiles," in *IEEE American Control Conference*, Seattle, WA, 2017, pp. 1091–1096.

[151] ——, "A posteriori multi-stage optimal trading under transaction costs and a diversification constraint," *arXiv preprint arXiv:1709.07527*, 2017, (Submitted).

[152] ——, "Reference trajectory planning under constraints and path tracking using linear time-varying model predictive control for agricultural machines," *Biosystems Engineering*, vol. 153, pp. 28–41, 2017.

[153] ——, "Stock trading via feedback control: Stochastic model predictive or genetic?" in *XVIII Workshop on Quantitative Finance (QFW2017), Poster presentation, arXiv preprint arXiv:1708.08857*, 2017.

[154] M. Graf Plessen, D. Bernardini, H. Esen, and A. Bemporad, "Multi-automated vehicle coordination using decoupled prioritized path planning for multi-lane one-and bi-directional traffic flow control," in *IEEE Conference on Decision and Control*, 2016, pp. 1582–1588.

[155] ——, "Spatial-based predictive control and geometric corridor planning for adaptive cruise control coupled with obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 38–50, 2018.

[156] M. Graf Plessen, P. Lima, J. Mårtensson, A. Bemporad, and B. Wahlberg, "Trajectory planning under vehicle dimension constraints using sequential linear programming," in *IEEE Conference on Intelligent Transportation Systems*, 2017, pp. 108–113.

[157] M. Graf Plessen, L. Puglia, T. Gabbriellini, and A. Bemporad, "Dynamic option hedging with transaction costs: A stochastic model predictive control approach," *International Journal of Robust and Nonlinear Control*, pp. 1–20, 2017.

[158] M. Graf Plessen, V. Semeraro, T. Wood, and R. Smith, "Optimization algorithms for nuclear norm based subspace identification with uniformly spaced frequency domain data," in *IEEE American Control Conference*, 2015, pp. 1119–1124.

[159] M. Graf Plessen, T. Wood, and R. Smith, "Nuclear norm minimization algorithms for subspace identification from non-uniformly spaced frequency data," in *IEEE European Control Conference*, 2015, pp. 2032–2037.

[160] ——, "Time-domain subspace identification algorithms using nuclear norm minimisation," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 903–908, 2015.

[161] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[162] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International Conference on Machine Learning*, 2014, pp. 1764–1772.

[163] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *IEEE American Control Conference*, 2012, pp. 4239–4244.

[164] A. Gray, M. Ali, Y. Gao, J. K. Hedrick, and F. Borrelli, "Semi-autonomous vehicle control for road departure and obstacle avoidance," *IFAC Control of Transportation Systems*, pp. 1–6, 2012.

[165] T. Gu, J. Snider, J. M. Dolan, and J. Lee, "Focused trajectory planning for autonomous on-road driving," in *IEEE IV*, 2013, pp. 547–552.

[166] T. Gu, J. Snider, J. M. Dolan, and J.-w. Lee, "Focused trajectory planning for autonomous on-road driving," in *IEEE IV*, 2013, pp. 547–552.

[167] E. Guizzo, "How googles self-driving car works," *IEEE Spectrum Online, October*, vol. 18, 2011.

[168] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*, 2012. [Online]. Available: http://www.gurobi.com

[169] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE ITS*, vol. 18, no. 6, pp. 1586–1595, 2017.

[170] S. Hallé, J. Laumonier, and B. Chaib-Draa, "A decentralized approach to collaborative driving coordination," in *IEEE Conference on Intelligent Transportation Systems*, 2004, pp. 453–458.

[171] M. Hamza and W. Anderson, "Soil compaction in cropping systems: a review of the nature, causes and possible solutions," *Soil and Tillage Research*, vol. 82, no. 2, pp. 121–145, 2005.

[172] J. Harrison and D. Kreps, "Martingales and arbitrage in multiperiod securities markets," *Journal of Economic Theory*, vol. 20, pp. 381–408, 1979.

[173] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[174] Y. Hattori, E. Ono, and S. Hosoe, "An optimum vehicle trajectory control for obstacle avoidance with the shortest longitudinal traveling distance," in *IEEE CMA*, 2008, pp. 13–20.

[175] J. Havlin, D. Kissel, L. Maddux, M. Claassen, and J. Long, "Crop rotation and tillage effects on soil organic carbon and nitrogen," *Soil Science Society of America Journal*, vol. 54, no. 2, pp. 448–452, 1990.

[176] W. He, G. Yan, and L. Da Xu, "Developing vehicular data cloud services in the iot environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1587–1595, 2014.

[177] E. Hebrard, E. OMahony, and B. OSullivan, "Constraint programming and combinatorial optimisation in numberjack," in *Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming*. Springer, 2010, pp. 181–185.

[178] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[179] L. Heilig, R. R. Negenborn, and S. Voß, "Cloud-based intelligent transportation systems using model predictive control," in *Computational Logistics*. Springer, 2015, pp. 464–477.

[180] I. Heller and C. Tompkins, "An extension of a theorem of Dantzigs," *Linear Inequalities and Related Systems*, vol. 38, pp. 247–254, 1956.

[181] F. Herzog, S. Keel, G. Dondi, L. M. Schumann, and H. P. Geering, "Model predictive control for portfolio selection," in *Proc. American Contr. Conf.*, Minneapolis, MN, 2006, pp. 1252–1259.

[182] F. Herzog, G. Dondi, and H. P. Geering, "Stochastic model predictive control and portfolio optimization," *International Journal of Theoretical and Applied Finance*, vol. 10, no. 02, pp. 203–233, 2007.

[183] S. L. Heston, "A closed-form solution for options with stochastic volatility with applications to bond and currency options," *Review of Financial Studies*, vol. 6, pp. 327–343, 1993.

[184] S. Heston and S. Nandi, "A closed-form GARCH option valuation model," *The Review of Financial Studies*, vol. 11, no. 3, pp. 585–625, 2000.

[185] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *IEEE ACC*, 2007, pp. 2296–2301.

[186] B. HomChaudhuri, A. Vahidi, and P. Pisu, "Fast model predictive control-based fuel efficient control strategy for a group of connected vehicles in urban road conditions," *IEEE CST*, vol. 25, no. 2, pp. 760–767, 2017.

[187] L. J. Hong and B. L. Nelson, "A brief introduction to optimization via simulation," in *IEEE Winter Simulation Conference*, 2009, pp. 75–85.

[188] B. Houska, H. J. Ferreau, and M. Diehl, "Acado toolkit–an open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[189] ——, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[190] J. Hull and A. White, "The pricing of options on assets with stochastic volatilities," *Journal of Finance*, vol. 42, no. 2, pp. 281–300, 1987.

[191] J. Hull, *Options, Futures, and Other Derivatives*, 6th ed. Upper Saddle River, NJ: Prentice Hall, 2006.

[192] R. Hult, G. R. Campos, E. Steinmetz, L. Hammarstrand, P. Falcone, and H. Wymeersch, "Coordination of cooperative autonomous vehicles: Toward safer and more efficient road transportation," *IEEE Signal Process. Mag.*, vol. 33, no. 6, pp. 74–84, 2016.

[193] R. Hussain, J. Son, H. Eun, S. Kim, and H. Oh, "Rethinking vehicular communications: Merging vanet with cloud computing," in *IEEE Conference on Cloud Computing Technology and Science*, 2012, pp. 606–609.

[194] IBM, Inc., *IBM ILOG CPLEX Optimization Studio 12.4 – User Manual*, 2012.

[195] ILOG, Inc., *CPLEX 11.0 User Manual*, Gentilly Cedex, France, 2008.

[196] P. Ioannou, C.-C. Chien *et al.*, "Autonomous intelligent cruise control," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 657–672, 1993.

[197] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[198] M. Jensen, D. Bochtis, and C. Sørensen, "Coverage planning for capacitated field operations, part ii: Optimisation," *Biosystems Engineering*, vol. 139, pp. 149–164, 2015.

[199] M. Jensen, D. Bochtis, C. Sørensen, M. Blas, and K. Lykkegaard, "In-field and inter-field path planning for agricultural transport units," *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 1054–1061, 2012.

[200] M. H. Jørgensen, "Agricultural field machinery for the future–from an engineering perspective," *Agronomy Research*, vol. 10, no. Special Issue 1, pp. 109–113, 2012.

[201] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[202] R. Kala and K. Warwick, "Planning autonomous vehicles in the absence of speed lanes using an elastic strip," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1743–1752, 2013.

[203] ——, "Dynamic distributed lanes: motion planning for multiple autonomous vehicles," *Applied intelligence*, vol. 41, no. 1, pp. 260–281, 2014.

[204] J. Kalmari, J. Backman, and A. Visala, "A toolkit for nonlinear model predictive control using gradient projection and code generation," *Control Engineering Practice*, vol. 39, pp. 56–66, 2015.

[205] M. A. S. Kamal, M. Mukai, J. Murata, and T. Kawabe, "On board eco-driving system for varying road-traffic environments using model predictive control," in *IEEE CCA*, 2010, pp. 1636–1641.

[206] Y. Kanayama and B. Hartman, "Smooth local path planning for autonomous vehicles," in *IEEE ICRA*, May 1989, pp. 1265–1270.

[207] Y. J. Kanayama and B. I. Hartman, "Smooth local-path planning for autonomous vehicles1," *The International Journal of Robotics Research*, vol. 16, no. 3, pp. 263–284, 1997.

[208] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *IEEE Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[209] J. Karlsson, N. Murgovski, and J. Sjöberg, "Temporal vs. spatial formulation of autonomous overtaking algorithms," in *IEEE ITSC*, 2016, pp. 1029–1034.

[210] K. Kato, "Bubbles in japan's stock markets: a macroeconomic analysis," 1995, working Paper, Columbia University.

[211] A. Katriniok and D. Abel, "Ltv-mpc approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 6828–6833.

[212] E. Kayacan, E. Kayacan, H. Ramon, and W. Saeys, "Learning in centralized nonlinear model predictive control: Application to an autonomous tractor-trailer system," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 197–205, 2015.

[213] ——, "Towards agrobots: Identification of the yaw dynamics and trajectory tracking of an autonomous tractor," *Computers and Electronics in Agriculture*, vol. 115, pp. 78–87, 2015.

[214] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager, "Optimal control of formula 1 race cars in a vdrift based virtual environment," in *Proceedings of the 18th IFAC World Congress*, vol. 18, 2011, pp. 11 907–11 912.

[215] D. P. Kelly and R. S. Sharp, "Time-optimal control of the race car: a numerical method to emulate the ideal driver," *Veh. Sys. Dyn.*, vol. 48, no. 12, pp. 1461–1474, 2010.

[216] H. Kern, "The resurgent japanese economy and a japan–united states free trade agreement," in *4th International Conference on the Restructuring of the Economic and Political System in Japan and Europe*. Singapore: World Scientific, 21–25 May 1996 1997, pp. 147–156.

[217] K. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1, pp. 307–319, 2003.

[218] K.-j. Kim and I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index," *Expert Systems with Applications*, vol. 19, no. 2, pp. 125–132, 2000.

[219] S. Kim, M. E. Lewis, and C. C. White, "Optimal vehicle routing with real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 178–188, 2005.

[220] P. Klaassen, "Financial asset-pricing theory and stochastic programming models for asset/liability management: A synthesis," *Management Science*, vol. 44, no. 1, pp. 31–48, 1998.

[221] R. C. Klemkosky and B. G. Resnick, "Put-call parity and market efficiency," *The Journal of Finance*, vol. 34, no. 5, pp. 1141–1155, 1979.

[222] D. Kogan and R. Murray, "Optimization-based navigation for the DARPA grand challenge," in *IEEE CDC*, 2006.

[223] I. Kolmanovsky, I. Siverguina, and B. Lygoe, "Optimization of powertrain operating policy for feasibility assesment and calibration: stochastic dynamic programming approach," in *Proc. American Contr. Conf.*, 2002, pp. 1425–1430.

[224] K. Komoriya and K. Tanie, "Trajectory design and control of a wheel-type mobile robot using B-spline curve," in *IEEE/RSJ Intern. Workshop on Intell. Rob. and Sys.*, September 1989, pp. 398–405.

[225] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *IEEE IV*, 2015, pp. 1094–1099.

[226] R. Korn, "Portfolio optimisation with strictly positive transaction costs and impulse control," *Finance and Stochastics*, vol. 2, no. 2, pp. 85–114, 1998.

[227] K. Korosec, "Elon musk says tesla vehicles will drive themselves in two years," *Fortune, December*, vol. 21, 2015.

[228] J. Koutník, J. Schmidhuber, and F. Gomez, "Online evolution of deep convolutional network for vision-based reinforcement learning," in *International Conference on Simulation of Adaptive Behavior*. Springer, 2014, pp. 260–269.

[229] R. Kouwenberg and T. Vorst, "Dynamic portfolio insurance: A stochastic programming approach," Econometric Institute and Department of Finance, Erasmus University, Rotterdam, Tech. Rep. 9909, 1998.

[230] D. Kouzoupis, H. Ferreau, H. Peyrl, and M. Diehl, "First-order methods in embedded nonlinear model predictive control," in *Proc. IEEE European Control Conference (ECC)*, Austria: Linz, 2015, pp. 2617–2622.

[231] T. Kraus, H. J. Ferreau, E. Kayacan, H. Ramon, J. De Baerdemaeker, M. Diehl, and W. Saeys, "Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles," *Computers and Electronics in Agriculture*, vol. 98, pp. 25–33, 2013.

[232] K. Kritayakirana and J. Gerdes, "Autonomous vehicle control at the limits of handling," *Intern. Jrnl. of Veh. Autonom. Sys.*, vol. 10, no. 4, pp. 271–296, 2012.

[233] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[234] D. Kuchin, P. Lega, A. Orlov, V. Koledov, and A. Irzhak, "The smallest and the fastest shape memory alloy actuator for micro-and nanorobotics," in *IEEE Conference on Manipulation, Automation and Robotics at Small Scales*, 2017, pp. 1–4.

[235] W. Kühn, *Fundamentals of road design*. WIT Press, 2013, vol. 20.

[236] H. D. Kutzbach, "Trends in power and machinery," *Journal of Agricultural Engineering Research*, vol. 76, no. 3, pp. 237–247, 2000.

[237] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, J. P. How *et al.*, "Motion planning for urban driving using RRT," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 1681–1686.

[238] L. Lamport, "Efficient algorithms for layer assignment problems," PhD thesis, University of Princeton, NJ, 1996.

[239] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[240] J. Larson, K.-Y. Liang, and K. H. Johansson, "A distributed framework for coordinated heavy-duty vehicle platooning," *IEEE ITS*, vol. 16, no. 1, pp. 419–429, 2015.

[241] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[242] S. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[243] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.

[244] S. Lefèvre, A. Carvalho, and F. Borrelli, "Autonomous car following: A learning-based approach," in *IEEE Intelligent Vehicles Symposium*, 2015, pp. 920–926.

[245] R. Lenain, B. Thuilot, C. Cariou, and P. Martinet, "Model predictive control for vehicle guidance in presence of sliding: application to farm vehicles path tracking," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Spain: Barcelona, 2005, pp. 885–890.

[246] ——, "High accuracy path tracking for vehicles in presence of sliding: Application to farm vehicle automatic guidance for agricultural tasks," *Autonomous Robots*, vol. 21, no. 1, pp. 79–97, 2006.

[247] ——, "Adaptive and predictive path tracking control for off-road mobile robots," *European Journal of Control*, vol. 13, no. 4, pp. 419–439, 2007.

[248] Z. Leng and M. Minor, "A simple tractor-trailer backing control law for path following," in *Proc. IEEE/RSJ Conference on Intelligent Robots and Systems*, Taiwan: Taipei, 2010, pp. 5538–5542.

[249] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Kolter, D. Langer, O. Pink, and V. Pratt, "Towards fully autonomous driving: Systems and algorithms," in *IEEE IV*, June 2011, pp. 163–168.

[250] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *IEEE Vehicular Technology Magazine*, vol. 2, no. 2, pp. 12–22, 2007.

[251] P. Li, M. Wendt, and G. Wozny, "Robust model predictive control under chance constraints," *Computers and Chemical Engineering*, vol. 24, no. 2-7, pp. 829–834, 2000.

[252] S. Li, K. Li, R. Rajamani, and J. Wang, "Model predictive multi-objective vehicular adaptive cruise control," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 556–566, 2011.

[253] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[254] P. F. Lima, M. Trincavelli, J. Martensson, and B. Wahlberg, "Clothoid-based model predictive control for autonomous driving," in *IEEE European Control Conference*, 2015, pp. 2983–2990.

[255] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[256] A. Lo, H. Mamaysky, and J. Wang, "Asset prices and trading volume under fixed transactions costs," National Bureau of Economic Research, Tech. Rep., 2001.

[257] M. S. Lobo, M. Fazel, and S. Boyd, "Portfolio optimization with linear and fixed transaction costs," *Annals of Operations Research*, vol. 152, no. 1, pp. 341–365, 2007.

[258] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[259] F. A. Longstaff and E. S. Schwartz, "Valuing american options by simulation: A simple least-squares approach," *Review of Financial Studies*, vol. 114, no. 1, pp. 113–147, 2001.

[260] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 7559–7565, 2014.

[261] T. Lozano-Perez, "Spatial planning: A configuration space approach," in *Autonom. robot veh.* Springer, 1990, pp. 259–271.

[262] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[263] J. Luck, S. Pitla, S. Shearer, T. Mueller, C. Dillon, J. Fulton, and S. Higgins, "Potential for pesticide and nutrient savings via map-based automatic boom section control of spray nozzles," *Computers and Electronics in Agriculture*, vol. 70, no. 1, pp. 19–26, 2010.

[264] A. Makhorin, *GLPK (GNU Linear Programming Kit) User's Guide*, 2004. [Online]. Available: http://www.gnu.org/software/glpk/glpk.html

[265] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[266] A. S. Marques, J. F. Audy, S. DAmours, and M. Rönnqvist, "Tactical and operational harvest planning," in *The Management of Industrial Forest Plantations*. Springer, 2014, pp. 239–267.

[267] H. Marzbani, R. Jazar, and M. Fard, "Better road design using clothoids," in *Sustainable Automotive Technologies 2014*. Springer Intern. Publish., 2015, pp. 25–40.

[268] C. M. Massera, M. H. Terra, and D. F. Wolf, "Guaranteed cost model predictive control-based driver assistance system for vehicle stabilization under tire parameters uncertainties," in *IEEE ITSC*, 2016, pp. 322–327.

[269] D. Q. Mayne, J. Rawlings, C. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[270] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[271] R. L. McDonald, *Derivatives Markets*. Pearson, 2013.

[272] L. G. McMillan, *McMillan on options*. John Wiley & Sons, 2011, vol. 229.

[273] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *IEEE ICRA*, 2011, pp. 4889–4895.

[274] P. Meindl and J. Primbs, "Dynamic hedging of single and multi-dimensional options with transaction costs: A general utility maximization approach," *Quantitative Finance*, vol. 8, no. 3, pp. 299–312, Apr. 2008.

[275] P. Meindl, "Portfolio optimization and dynamic hedging with receding horizon control, stochastic programming, and Monte Carlo simulation," Ph.D. dissertation, Dept. Management Science & Engineering, Stanford University, 2006.

[276] R. Merton, "The theory of rational option pricing," *Bell Journal of Economics and Management Science*, vol. 4, pp. 141–183, 1973.

[277] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[278] C. W. Misner, Ed., *Gravitation*. San Francisco, CA: Freeman, 1973, ch. Efficient algorithms for layer assignment problems.

[279] M. N. Mladenovic and M. Abbas, "Priority-based intersection control framework for self-driving vehicles: Agent-based model development and evaluation," in *International Conference on Connected Vehicles and Expo*, 2014, pp. 377–384.

[280] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[281] R. Möbus, M. Baotic, and M. Morari, *Multi-object adaptive cruise control*. Springer, 2003.

[282] M. Montemerlo *et al.*, "Junior: The Stanford entry in the urban challenge," *JFR*, vol. 25, no. 9, pp. 569–597, 2008.

[283] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[284] S. J. Moorehead, C. K. Wellington, B. J. Gilmore, and C. Vallespi, "Automating orchards: A system of autonomous tractors for orchard maintenance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Agricultural Robotics, Vilamoura, Portugal*, 2012.

[285] A. J. Morton and S. R. Pliska, "Optimal portfolio management with fixed transaction costs," *Mathematical Finance*, vol. 5, no. 4, pp. 337–356, 1995.

[286] D. Muñoz de la Peña, A. Bemporad, and T. Alamo, "Stochastic programming applied to model predictive control," in *Proc. 44th IEEE Conf. on Decision and Control and European Control Conf.*, Sevilla, Spain, 2005, pp. 1361–1366.

[287] M. Nanao and T. Ohtsuka, "Vehicle dynamics control for collision avoidance considering physical limitations," in *Proc. of Soc. of Instrum. and Ctrl Eng. conf.* IEEE, 2011, pp. 688–693.

[288] S. Natenberg, *Option volatility and pricing: advanced trading strategies and techniques*. McGraw Hill Professional, 2014.

[289] National Highway Traffic Safety Administration, "Traffic safety facts, 2014: a compilation of motor vehicle crash data from the fatality analysis reporting system and the general estimates system. dot hs 812261," *Department of Transportation, Washington, DC*, 2014.

[290] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[291] W. Nelson, "Continuous-curvature paths for autonomous vehicles," in *IEEE ICRA*, May 1989, pp. 1260–1264.

[292] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 969–996, 2006.

[293] M. Neumann, "Parallel grasp with path-relinking for job shop scheduling," *Mol. Phys.*, vol. 50, no. 2, pp. 841–843, 1983.

[294] J. Nilsson and J. Sjoberg, "Strategic decision making for automated driving on two-lane, one way roads using model predictive control," in *IEEE Intelligent Vehicles Symposium*, 2013, pp. 1253–1258.

[295] J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg, "Receding horizon maneuver generation for automated highway driving," *CEP*, vol. 41, pp. 124–133, 2015.

[296] Nvidia, "Tesla P100," https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-PCIe-datasheet.pdf, 2016.

[297] J. Oksanen, Timo; Backman, "Standardization proposal on implement guidance for ISO 11783 compatible tractor-implement systems," 2015. [Online]. Available: http://urn.fi/URN:ISBN:978-952-60-6165-8

[298] T. Oksanen and A. Visala, "Optimal control of tractor-trailer system in headlands," in *Proc. ASAE Conference on Automation Technology for Off-Road Equipment (ATOE)*, Japan: Kyoto, 2004, pp. 255–263.

[299] A. Orfanou, P. Busato, D. Bochtis, G. Edwards, D. Pavlou, C. Sørensen, and R. Berruto, "Scheduling for machinery fleets in biomass multiple-field operations," *Computers and Electronics in Agriculture*, vol. 94, pp. 12–19, 2013.

[300] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE T-IV*, vol. 1, no. 1, pp. 33–55, 2016.

[301] R. Palmer, D. Wild, and K. Runtz, "Improving the efficiency of field operations," *Biosystems Engineering*, vol. 84, no. 3, pp. 283–288, 2003.

[302] L. E. Parker, "Path planning and motion coordination in multiple mobile robot teams," *Encyclopedia of complexity and system science*, pp. 5783–5800, 2009.

[303] G. Pataki, "Teaching integer programming formulations using the traveling salesman problem," *SIAM Review*, vol. 45, no. 1, pp. 116–123, 2003.

[304] F. W. Patel, *Title of Book*, ser. Monographs on Technical Aspects. New York: Dover, 2002, vol. II.

[305] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 18–33, 2014.

[306] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov, "Combining neural networks and tree search for task and motion planning in challenging environments," *arXiv preprint arXiv:1703.07887*, 2017.

[307] M. Pivtoraiko, R. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *JFR*, vol. 26, no. 3, pp. 308–333, 2009.

[308] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, 1989, pp. 305–313.

[309] J. A. Primbs, "Stochastic receding horizon control of constrained linear systems with state and control multiplicative noise," in *Proc. American Contr. Conf.*, New York, NY, 2007, pp. 4470–4475.

[310] ——, "LQR and receding horizon approaches to multi-dimensional option hedging under transaction costs," in *Proc. American Contr. Conf.*, 2010, pp. 6891–6896.

[311] J. A. Primbs and C. H. Sung, "Stochastic receding horizon control of constrained linear systems with state and control multiplicative noise," *IEEE Trans. Automatic Control*, vol. 54, no. 2, pp. 221–230, 2009.

[312] J. A. Primbs and Y. Yamada, "A new computational tool for analyzing dynamic hedging under transaction costs," *Quantitative Finance*, vol. 8, no. 4, pp. 405–413, 2008.

[313] J. Primbs, "A soft constraint approach to stochastic receding horizon control," in *Proc. 46th IEEE Conf. on Decision and Control*, 2007, pp. 4797–4802.

[314] J. A. Primbs, "Dynamic hedging of basket options under proportional transaction costs using receding horizon control," *International Journal of Control*, vol. 82, no. 10, pp. 1841–1855, 2009.

[315] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, ser. Monographs on Technical Aspects. John Wiley & Sons, 2005.

[316] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An miqp perspective," in *IEEE ITSC*, 2016, pp. 205–210.

[317] R. Rajamani, *Vehicle Dynamics and Control*, ser. Mechanical Engineering Series. Springer US, 2011.

[318] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping," in *International Conference on Machine Learning*, 1998, pp. 463–471.

[319] R. Raper, "Agricultural traffic impacts on soil," *Journal of Terramechanics*, vol. 42, no. 3, pp. 259–280, 2005.

[320] J. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Systems Magazine*, pp. 38–52, Jun. 2000.

[321] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[322] R. Reghelin and L. V. Arruda, "Optimizing coordinated motion planning for multiple car-like robots on a segment of highway," *Robotica*, pp. 1–17.

[323] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz, "Cartalk 2000: Safe and comfortable driving based upon inter-vehicle-communication," in *IEEE Intelligent Vehicle Symposium*, vol. 2, 2002, pp. 545–550.

[324] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value at risk," *Journal of Risk*, vol. 2, pp. 21–42, 2000.

[325] D. Sabelhaus, F. Röben, L. P. M. zu Helligen, and P. S. Lammers, "Using continuous-curvature paths to generate feasible headland turn manoeuvres," *Biosystems Engineering*, vol. 116, no. 4, pp. 399–409, 2013.

[326] SAE, "Automated driving – levels of driving automation are defined in new SAE international standard J3016," https://techcrunch.com/2017/08/03/ cadillacs-super-cruise-autopilot-is-ready-for- the-expressway/, 2014.

[327] T. Salimans, J. Ho, X. Chen, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[328] A. Scheuer and T. Fraichard, "Planning continuous-curvature paths for car-like robots," in *Proc. IEEE/RSJ Conference on Intelligent Robots and Systems*, Japan: Osaka, 1996, pp. 1304–1311.

[329] G. Schildbach, M. Soppert, and F. Borrelli, "A collision avoidance system at intersections using robust model predictive control," in *IEEE IV*, 2016, pp. 233–238.

[330] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *IEEE European Control Conference*, 2001, pp. 2603–2608.

[331] T. Schouwenaars, B. Mettler, E. Feron, and J. P. How, "Robust motion planning using a maneuver automation with built-in uncertainties," in *IEEE American Control Conference*, vol. 3, 2003, pp. 2211–2216.

[332] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[333] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[334] R. N. Sengupta and R. Kumar, "Robust and reliable portfolio optimization formulation of a chance constrained problem," *Foundations of Computing and Decision Sciences*, vol. 42, no. 1, pp. 83–117, 2017.

[335] H. Seyyedhasani and J. S. Dvorak, "Using the vehicle routing problem to reduce field completion times with multiple machines," *Computers and Electronics in Agriculture*, vol. 134, pp. 142–150, 2017.

[336] A. Sharda, J. P. Fulton, T. P. McDonald, and C. J. Brodbeck, "Real-time nozzle flow uniformity when using automatic section control on agricultural sprayers," *Computers and Electronics in Agriculture*, vol. 79, no. 2, pp. 169–179, 2011.

[337] N. Shchetko, "Laser eyes pose price hurdle for driverless cars," *The Wall Street Journal*, vol. 21, 2014.

[338] S. E. Shladover, "Cooperative (rather than autonomous) vehicle-highway automation systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 1, no. 1, pp. 10–19, 2009.

[339] M. L. Sichitiu and M. Kihl, "Inter-vehicle communication systems: a survey," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 2, pp. 88–105, 2008.

[340] H. T. Siegelmann and E. D. Sontag, "Turing computability with neural nets," *Applied Mathematics Letters*, vol. 4, no. 6, pp. 77–80, 1991.

[341] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning*, 2014, pp. 387–395.

[342] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[343] R. Solea and U. Nunes, "Trajectory planning with velocity planner for fully-automated passenger vehicles," in *IEEE ITSC*, 2006, pp. 474–480.

[344] Y. Song and J. W. Grizzle, "The extended kalman filter as a local asymptotic observer for nonlinear discrete-time systems," in *IEEE American Control Conference*. IEEE, 1992, pp. 3365–3369.

[345] C. Sørensen, S. Fountas, E. Nash, L. Pesonen, D. Bochtis, S. M. Pedersen, B. Basso, and S. Blackmore, "Conceptual model of a future farm management information system," *Computers and Electronics in Agriculture*, vol. 72, no. 1, pp. 37–47, 2010.

[346] Statistisches Bundesamt Deutschland, "Land- und Forstwirtschaft," https://www.destatis.de/DE/Publikationen/StatistischesJahrbuch/LandForstwirtschaft.pdf?__blob=publicationFile, Oct. 2016.

[347] H. R. Stoll and R. E. Whaley, *Futures and options: theory and applications*. Southwestern Publishing, 1993.

[348] Y. Sun, G. Aw, R. Loxton, and K. L. Teo, "Chance-constrained optimization for pension fund portfolios in the presence of default risk," *European Journal of Operational Research*, vol. 256, no. 1, pp. 205–214, 2017.

[349] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[350] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, 2000, pp. 1057–1063.

[351] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, 2001.

[352] TechCrunch, "Cadillacs super cruise autopilot is ready for the expressway," https://techcrunch.com/2017/08/03/cadillacs-super-cruise-autopilot-is-ready-for- the-expressway/, 2017.

[353] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[354] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA grand challenge," *JFR*, vol. 23, no. 9, pp. 661–692, 2006.

[355] B. Thuilot, C. Cariou, P. Martinet, and M. Berducat, "Automatic guidance of a farm tractor relying on a single CP-DGPS," *Autonomous Robots*, vol. 13, no. 1, pp. 53–71, 2002.

[356] X. Tian, K. Benkrid, and X. Gu, "High performance Monte-Carlo based option pricing on FPGAs," *Engineering Letters*, vol. 16, no. 3, pp. 434–442, 2008.

[357] P. Toth and D. Vigo, *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[358] Trafikanalys, "Automatiserad kolonnkörning – en lösning för framtiden?" https://www.trafa.se/globalassets/rapporter/2016/rapport-2016_22-automatiserad-kolonnkorning---en-losning-for-framtiden.pdf, 2016.

[359] S. Ulbrich and M. Maurer, "Towards tactical lane change behavior planning for automated vehicles," in *IEEE Conference on Intelligent Transportation Systems*, 2015, pp. 989–995.

[360] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *JFR*, vol. 25, no. 8, pp. 425–466, 2008.

[361] C. F. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Transactions on Automatic Control*, vol. 23, no. 3, pp. 395–404, 1978.

[362] F. B. Veliz, J.-P. Watson, A. Weintraub, R. J.-B. Wets, and D. L. Woodruff, "Stochastic optimization models in forest planning: a progressive hedging solution approach," *Annals of Operations Research*, vol. 232, no. 1, pp. 259–274, 2015.

[363] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[364] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear mpc in real-time," in *IEEE CDC*, 2014, pp. 2505–2510.

[365] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking with geometric continuous-curvature path planning," in *IEEE IV*, 2014, pp. 465–471.

[366] S. G. Vougioukas, "A distributed control framework for motion coordination of teams of autonomous agricultural vehicles," *Biosystems engineering*, vol. 113, no. 3, pp. 284–297, 2012.

[367] W. Wang and S. Boyd, "Fast model predictive control using online optimization," in *Proc. 17th IFAC World Congress*, 2008, pp. 6974–6979.

[368] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[369] M. Werling and D. Liccardo, "Automatic collision avoidance using model-predictive online optimization," in *IEEE CDC*, 2012, pp. 6309–6314.

[370] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *IEEE ICRA*, 2010, pp. 987–993.

[371] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014.

[372] H. P. Williams, *Model building in mathematical programming*. John Wiley & Sons, 2013.

[373] P. Wilmott, *On Quantitative Finance*, 2nd ed. West Sussex PO19 8SQ, England: Wiley and Sons, 2006.

[374] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *arXiv preprint arXiv:1612.01079*, 2016.

[375] J. Xu, B. L. Nelson, and J. Hong, "Industrial strength compass: A comprehensive algorithm and software for optimization via simulation," *ACM Transactions on Modeling and Computer Simulation*, vol. 20, no. 1, p. 3, 2010.

[376] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 2061–2067.

[377] P. C. Young, *Recursive estimation and time-series analysis: an introduction*. Springer Science & Business Media, 2012.

[378] S. K. Zegeye, B. De Schutter, H. Hellendoorn, and E. Breunesse, "Reduction of travel times and traffic emissions using model predictive control," in *IEEE American Control Conference*, 2009, pp. 5392–5397.

[379] K. Zhou, A. L. Jensen, D. D. Bochtis, and C. G. Sørensen, "Quantifying the benefits of alternative fieldwork patterns in a potato cultivation system," *Computers and Electronics in Agriculture*, vol. 119, pp. 228–240, 2015.

[380] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha–A local, continuous method," in *IEEE Intelligent Vehicles Symposium*, 2014, pp. 450–457.