IMT School for Advanced Studies, Lucca

Lucca, Italy

The Capacitated Spanning Forest Problem

PhD Program in Complex Networks

XXIX Cycle

By

George Davidescu

2019

The dissertation of George Davidescu is approved.

Program Coordinator: Prof. Guido Caldarelli, IMT School for Advanced Studies Lucca

Supervisor: Professor Guido Caldarelli, IMT School for Advanced Studies Lucca

IMT School for Advanced Studies, Lucca

2019

Acknowledgements

My gratitude goes to my supervisor Professor Guido Caldarelli for his unwavering support and help in bringing this work to fruition. Special thanks to Professor Valeriy Vyatkin of Aalto University and Luleå Technical University for his invitation, supervision, and support during an Erasmus visit. My thanks go also to Professor Thomas Stützle of Université Libre de Bruxelles and Professor Andrey Filchenkov of ITMO University for their collaboration on the algorithms described in this work. My gratitude also goes to the reviewers Professor Alfonso Damiano and Professor Antonio Scala for their valuable comments and suggestions. A big thank you to Angelo Facchini for his useful feedback on the contents of this thesis.

The author acknowledges the computational resources provided by the Aalto Science-IT project that were used for performing the simulations in Chapters 5 and 6.

1. Chapter 5 is based on the paper "Network planning in smart grids via a local search heuristic for spanning forest problems"(DSV17), published in the proceeding of the IEEE 26th International Symposium on Industrial Electronics (ISIE) and co-authored with Thomas Stützle and Valeriy Vyatkin.
2. Chapter 6 is based on the paper "Network planning and self-repair in models of urban distribution networks via hill climbing."(DV17), published in the proceeding of the 43rd Annual Conference of the IEEE Industrial Electronics Society and co-authored with Valeriy Vyatkin.

3. Chapter 7 is based on the paper "A flow-based heuristic algorithm for network operations planning in smart grids"(DFMV18), published in the proceeding of the 44th Annual Conference of the IEEE Industrial Electronics Society and co-authored with Amir Muratov, Andrey Filchenkov and Valeriy Vyatkin.

Contents

List of Figures

List of Tables

Publications

1. G. Davidescu, A. Filchenkov, A. Muratov, and V. Vyatkin. A flow-based heuristic algorithm for network operations planning in smart grids. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, pages 3529–3534, Oct 2018.
2. G. Davidescu and V. Vyatkin. Network planning and self-repair in models of urban distribution networks via hill climbing. In *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 5477–5482, Oct 2017.
3. G. Davidescu, T. Stützle, and V. Vyatkin. Network planning in smart grids via a local search heuristic for spanning forest problems. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pages 1212–1218, June 2017.
4. A. Bessi, M. Coletto, G. Davidescu, A. Scala, G. Caldarelli, W. Quattrociocchi. Science vs conspiracy: Collective narratives in the age of misinformation. *PloS one*, 10(2), e0118093, Feb 2015.
5. A. Bessi, M. Coletto, G. Davidescu, A. Scala, W. Quattrociocchi. Misinformation in the loop: the emergence of narratives in Online Social Networks. *itAIS 2014*, Nov 2014.

Presentations

1. Davidescu, G. Heuristic Algorithms for the Capacitated Spanning Forest Problem with a Smart Grid Application. 9th International Conference on Complex Systems, Cambridge, USA, July 26, 2018.
2. Davidescu, G. Network Planning and Self-Repair in Models of Urban Distribution Networks. Seminar for the Industrial Informatics and Automation group at Luleå Technical University, Luleå, Sweden, Nov 27, 2017.
3. Davidescu, G. Support Vector Machines Applied to Facebook Sentiment Analysis. Seminar delivered in the Machine Learning seminar series of the Industrial Technologies in Automation group at Aalto University, Helsinki, Finland, April 17, 2017.
4. Davidescu, G. New Developments on Self-Healing in Smart Grids. Seminar delivered in the postgraduate seminar series at Aalto University, Helsinki, Finland, April 5, 2017.
5. Davidescu, G. Self-Healing in Smart Grids. Seminar delivered in the annual meeting of the Smart Control Architecture for Smart Grids (SAGA) project, Viljandi, Estonia, Dec 15, 2016.
6. Bessi, A., Coletto, M., Davidescu, G. A., Scala, A., Caldarelli, G., Quattrociocchi, W. Sensing information-based communities in the age of misinformation. European Conference on Complex Systems 2014, Lucca, Italy, Sep 2014.
7. Davidescu, G. Accelerated Dataflow Programming using the MaxCompiler. Seminar delivered at the Max Planck Institute for Molecular Genetics, Berlin, Germany, May 21, 2013.
8. Davidescu, G. RegAlign: An Algorithm for the Alignment of Gene Regulatory Networks. Poster presented at the *3rd ISCB Student Council Symposium*, Vienna, Austria, July 2007.

Abstract

The smart grid is envisioned as a reconfigurable energy exchange network that is resilient to failures. An expected feature of the future smart grid is optimal power distribution from energy producers to consumers, also referred to as network planning. This entails allocating finite energy resources to customers in order to optimally satisfy all customer demands, subject to constraints on the topology of the graph. This thesis deals with modeling this problem as the CAPACITATED SPANNING FOREST PROBLEM (CSF), namely the optimization problem of creating a spanning forest with a capacity constraint on each tree limiting its total weight. I prove the NP-completeness of this problem and provide three different heuristic algorithms for solving it: a Local Search heuristic, a Hill Climbing heuristic, and a Max Flow-based heuristic, each of which has been published in a peer-reviewed IEEE conference. These are the first algorithmic approaches in the literature addressing this problem. Each algorithm is an improvement over the previous in solution quality and efficiency. Using a simulation-based approach I empirically demonstrate the suitability of these algorithms for planning the initial configuration of a smart grid's topology, and for recovery when faults occur.

Chapter 1

Background on Smart Grids

1.1 Motivation

Electrification was rated "the greatest engineering achievement of the 20th century" by the United States National Academy of Engineering (Ros10), taking precedence over the automobile, the airplane, the Internet, and highways. Electric power and the infrastructure that supplies it are vital to modern human life, and serve as the foundation for essentially all industries. The importance of its continuous operation and reliability cannot be overstated.

The development of new technology and recent industrial, social, and policy factors are leading toward a reconfiguration of the existing energy infrastructure. The demand for energy in the world is growing faster than the growth rate of energy generation (LDS10). World energy consumption is projected to increase by 44% from 2006 to 2030. In addition, in recent years power outages have become increasingly frequent in the United States (Col10), and outages are often discovered only after they are reported by consumers (GWP$^+$14). The loss of electric power today has a more profound effect than in the past due to our reliance on digital electronic technologies.

A White House report entitled "Economic Benefits of Increasing Electric Grid Resilience to Weather Outages" (Exe13) found that weather-

related outages in the period 2003-2012 are estimated to have cost the U.S. economy an average of $18 billion to $33 billion annually. The report concluded that continued investment in grid modernization and resilience will mitigate these costs, saving billions of dollars and reducing the human hardship caused by extreme weather.

There is a growing need to deliver higher quality and more reliable power from an aging infrastructure while keeping costs low. Matching generation to demand is difficult because energy producers do not have adequate methods to predict demand and to perform demand reduction (know as load shedding). As a result, they over-generate power during periods of peak demand, which is expensive and environmentally unsound. Similarly, the difficulty of predicting volatile generation such as wind and solar power makes these energy sources challenging to integrate into the electric grid (KBD+12).

Information and communication technologies are currently being integrated into traditional energy delivery systems. There is also a worldwide focus on "green" renewable energy and climate change. The use of renewable energy requires a grid that can cope with bi-directional flows and the unpredictability of renewable sources, as well as the incorporation of distributed storage. Finally, there is also an increasing desire from consumers for a role in energy production and management as "prosumers".

These needs for change call for the transformation of the electric power system from a "mostly unidirectional, centralized, and hierarchical organization into a distributed, networked, and automated energy value chain" (Ros10). In response to these needs, the United States National Institute of Standards and Technology (NIST) spurred national efforts to develop the next-generation electric power system, commonly referred to as the "Smart Grid" (GWP+14).

The Smart Grid is an enhancement of the 20th century power grid that uses automation and Information and Communication Technology (ICT) to better manage energy production and distribution. Whereas traditional power grids supply energy from a few central generators to a large number of customers in a mainly static configuration, the Smart

Grid is an automated and distributed energy delivery network that allows for bi-directional energy flows between prosumers, dynamic routing, flexible loads, and variable tariffs.

However, because of their complexity the vulnerability of smart grids must also be addressed. As they are a critical infrastructure, it is crucial that smart grids be reliable, robust and resilient to failures, whether accidental or malicious. The ability to autonomously "self-heal" in the event of failure is expected to be an important characteristic of smart grids.

It is this challenge of resilience in smart grids that is the principal motivation behind this work.

It is noteworthy that although grid digitization is still in incipient in most of the United States, some smart grid pilot projects at the Medium-Voltage level featuring some degree of automation have already been implemented in Europe and other areas of the world, where Italy is leading the field with ongoing projects deployed after 2010. Enel's Telegestore project is regarded as the first commercial scale use of smart grid smart metering application in consumers' homes (TA16), while seven smart grid pilot projects from six operators (Enel, ACEA, A2A, ASM, A.S.SE.M., Deval) have been deployed on Medium Voltage networks in Italy in the early 2010s. These projects implement a selection of features such as demand response, electric vehicle charging, monitoring and diagnosis of components such as circuit breakers, network planning, automated fault detection and localization, and protection mechanisms (CPP$^+$15).

In particular, the A2A project (DFOP15) implemented in Milan proposes a novel distributed automated method for selective fault detection based on logic selectivity (a term for simple procedures triggered by inter-substation communications) and rapid network reconfiguration; however, at the last time of publication the method had still not been completely tested and its performance was only estimated. The pilot project implemented by A.S.SE.M. on a Medium Voltage network in central Italy also featured fault isolation through logic selectivity (DFFM15). Nevertheless, reconfiguring the topology of the electrical network in response to faults remains a complex problem, and algorithms are still be-

ing developed to do so optimally; (PPL+16) propose a genetic algorithm approach for reconfigurations of the ACEA network.

Other countries in which smart grid demonstration projects and smart grid technology rollouts have been performed include Korea, Japan, Australia, the United Kingdom, the United States, Spain and Portugal (TA16).

1.2 Description of the electrical infrastructure

The electric power network, power grid, or "grid", is a physical system that delivers electrical energy from production facilities (generators) to consumers. It is comprised of a networked infrastructure that spans hundreds of kilometers and connects intermediary substations by power lines (arcs).

The present infrastructure of the electric grid is composed of these major domains:

Generation: Produces electric energy in various ways at medium and low voltages. Currently, 62% of worldwide energy generation comes from coal and gas, 13% from nuclear power sources, 16% from hydroelectric sources, and less than 4% from other renewable sources of energy (Ros10).

Transmission: Carries electricity over long distances via High-Voltage (HV) (>200 kV) infrastructure. Transmission networks have a meshed structure for redundancy (see Figure 1).

Distribution: Lowers voltage and distributes it for consumption. Medium-Voltage (MV) (10kV-60kV) infrastructure serves industry while Low-Voltage (LV) (220V-380V) serves residential and commercial areas. The MV infrastructure is connected to the HV and LV infrastructures, while the LV infrastructure is connected to the MV infrastructure. Distribution networks typically have a planar and radial (tree-like) structure to facilitate metering and reduce costs.

Consumption: Energy is used by consumers in a multitude of ways. Energy needs are typically less than 20 kW for a residence, 20-200

kW for commercial buildings, and >200kW for industrial sites.

Operations: Manages the supply-side movement of electricity and is responsible for the smooth operation and maintenance of the power system.

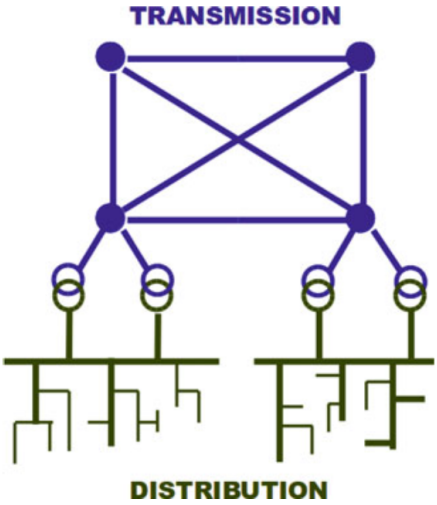


Figure 1: Representation of the electrical power system, consisting of Transmission (High voltage lines in blue) and Distribution (Medium voltage and Low voltage in green) networks. Transmission networks dispatch power over long distances and have a meshed structure for robustness. Distribution networks are local and have mostly a radial (tree-like) structure to reduce economic costs and to simplify the per-user calculation of power consumption. Industries are generally served by medium voltage, while residential customers are served by low voltage. Reproduced from (SCC$^+$14).

1.2.1 Description of the distribution network

The following is a description of the main components in the distribution network, based on the one by (DCI13), to familiarize the reader with the nature of the grid and its relevant terminology. The Medium and Low-voltage distribution infrastructures consist of the following categories of

components:

Substations: Substations are facilities that direct electricity across the network and control its voltage. They are equipped with switches, circuit breakers, transformers, and control equipment. They also connect the MV to the HV and LV infrastructures, lowering the voltage at each step, and connect generators to the grid. They are generally unmanned and rely on control systems such as Supervisory Control and Data Acquisition (SCADA) for supervision and control.

Power (or generating) **stations:** Supply energy to the distribution network from a variety of sources, including chemical combustion, nuclear fission, flowing water, wind, solar radiation, and geothermal heat. Generators can be classified as:

- Bulk generators: Classical generators that generate energy in non-intermittent large quantity for transport over long distances.
- Distributed Energy Resources (DERs): Produce energy in smaller quantities and are located close to the load they serve. Typically produce energy from renewable energy sources (wind, solar, small hydro, biomass, geothermal).
- Micro or mini-generators: Intermittent small-scale generators in the low-voltage network that are owned by a customer.

Distributed storage: Function as storage for DERs. Can be used to smooth the power generation profile of intermittent renewable generators, or can be used by Distributed System Operators (DSOs) to enhance network performance, helping supply power during peak operation and improving quality of service. To this end, they can store energy during off-peak periods or from intermittent sources, and sell it back on the energy market during peak periods, generating a profit from the price margin.

Loads: A load is anything connected to the grid that extracts energy from the distribution network.

- Non-flexible loads: Classical loads that cannot tolerate a loss of power. Non-flexible loads in the low-voltage infrastructure include households, small enterprises, and hospitals.
- Flexible loads: Loads that offer flexibility in the power profile, e.g. shifting loads to less expensive time slots, changing the amount of energy or changing the tariff. Examples include smart appliances and charging stations for electric vehicles.

Power lines, or arcs connecting components in the network. Power lines have protection units such as breakers or an associated switch.

Switches and circuit breakers: Modify the topology of the network when they are tripped (in the case of circuit breakers) or alternatively opened or closed (in the case of switches). Circuit breakers cut off flow to an arc, while switches can be activated or deactivated to route power flow.

A combination of the above items is also possible. For example, a residential consumer that produces energy (a.k.a. a "prosumer") can be both a generator and a load.

The electrical infrastructure can be represented as a network, or graph. A vertex in the graph represents a power station, a load, or a substation, or a combination thereof, while an edge represents a power line. The topology of the High-Voltage network typically follows a meshed configuration. The topology of the Medium-Voltage network is typically a radial graph or a partially meshed graph for redundancy. The Low-Voltage electric infrastructure is typically a radial or planar tree graph, or a loop or ring graph with a few redundant links.

Paraphrasing (GWP$^+$14), "historically, distribution systems have had radial configurations and little telemetry, almost all communications within the infrastructure being performed by humans. The primary sensor in this situation is the customer with a telephone, whose call initiates the dispatch of a field crew to restore power".

Although communications interfaces within the distribution domain have traditionally been hierarchical and unidirectional, they are adapting

to work in both directions, transmitting information from sensors, just as electrical connections are beginning to support bi-directional flow.

1.2.2 Smart Grid features

The principal feature differentiating the smart grid from the traditional power network is pervasive automation facilitated by digital Information & Communication Technologies, which allows for a number of system-wide features of the grid. Whereas the traditional power grid relies on electromechanical technology and has little internal regulation or means for communication between devices, the smart grid employs digital technology that enables increased communication between devices and facilitates remote control and self-regulation. The Distribution domain in the smart grid will be automated and will communicate in a more granular fashion in real-time with the Operations domain to manage power flows, including information from energy markets and other environmental and security factors.

On a traditional power grid lacking digitization an electric utility has few means to control or manage the system other than matching generation to consumption. In the smart grid, sensors are pervasive and digital technology allows for self-monitoring and pervasive control of the grid. The smart grid is reconfigurable, enabling automatic changes in the topology of the network in the case of failures, thus maintaining service and expediting system recovery. This type of operation is currently performed manually on traditional grids. The automated processing for reconfiguration can be implemented in a distributed rather than centralised fashion, with each switch reacting independently to its sensory inputs. This local control of local topology includes the adoption of self-healing algorithms (SMRP11). Distribution Transformer Controllers enable the optimal operation of the local network based on constantly updated distribution system parameters that optimize energy flows, network topology and offer self-healing algorithms in collaboration with primary substation automation (SMRP11).

The smart grid will also feature a dynamic energy market enabled

by digital technology. Constant updates in energy prices to consumers from the energy market enable demand response, which entails consumers using or storing energy during periods when tariffs are lower, and can also entail selling energy from storage back on the grid when it is more expensive. With demand response consumers will have the ability to dynamically change their electricity consumption in response to price signals from the grid, which in turn reflect total user demand. The smart grid system also enables the integration of intermittent renewable energy sources and distributed storage, selling their energy back on the grid. This can even be done from electric vehicles, which function as mobile distributed storage. To facilitate this, the distribution network must adapt to bi-directional flows rather than conventional unidirectional flows, as customers may also generate, store, and manage the use of energy.

The smart grid enables a new market of service providers - sellers of smart sensors, electric vehicles, and prosumer technologies, increasing consumer choice. The smart grid will provide the same kind of foundational infrastructure for a new business ecosystem in the same way that the Internet did for new online business models, such as crowdfunding or online marketplaces. Service providers in the smart grid are organizations or individuals providing services to electrical customers and to utilities. These entities will create new and innovative services and products to meet the requirements and opportunities presented by the evolving smart grid. For example, service providers may provide the interface enabling the customer to interact with the energy market. Choice is also reflected in energy providers as customers can more easily select green resource or prosumers according to their preferences, rather than being tied to a single utility.

Further anticipated benefits of the smart grid, elaborated by the National Institute of Standards and Technology (NIST) (GWP+14), include:

- "Improved power reliability and quality
- Optimized use of existing power facilities and networks

- Averts construction of back-up power plants during peak operation
- Improved resilience to disruption
- Easier deployment of renewable energy sources
- Automated maintenance and operation
- Reduced greenhouse gas emissions
- Reduced fossil fuel consumption, by reducing the need for inefficient generation during peak usage periods
- Enables the transition to plug-in electric vehicles and new energy storage options"

1.3 Smart Grid challenges

There are many open challenges in the field of smart grids. These can be summarized as challenges dealing with 1) the operation of the physical electric grid infrastructure of the smart grid, 2) the communication infrastructure of the grid, and 3) the protection of these two components (FMXY12). These challenges are outlined in the next section. I will briefly describe the open problems concerning smart grids as a whole so the reader is aware of them, and then focus on problems in the domain of protection and resilience.

The problems of resilience are the ones most relevant to this thesis, since they motivated the central question addressed by it. The reconfiguration mechanisms used to address resilience are exploited to address the thesis problem, introduced in Chapter 2. This work is concerned both with utilizing the reconfigurability of smart grids to optimally balance loads and resources within tight bounds, as well as exploiting the same reconfigurability to restore connectivity following disruptions.

1.3.1 Issues of the power grid infrastructure

Integrating intermittent renewable energy sources and distributed storage

Generation in the smart grid includes both traditional generation sources and distributed energy resources (DER). Traditional generation sources include coal, nuclear, and large-scale hydro generation usually attached to the transmission domain. DER refers to prosumer generation, storage and intermittent sources of energy.

The types of generation in the smart grid can be classified as:

1. Renewable and intermittent: Wind, solar

2. Renewable and non-intermittent: Hydro, biomass, geothermal, pumped storage

3. Non-renewable and non-intermittent: Nuclear, coal, gas

It is predicted that distributed, renewable energy sources such as low-cost wind and solar energy generators will be widely used in smart grids in the future (FMXY12). However, these sources provide power in a volatile, intermittent manner, as solar and wind energy is sensitive to intermittent weather conditions. The grid, however, operates from a "power-on-demand" paradigm for which volatile sources are ill-suited.

Integrating variable resources is one of the challenges of smart grids. This is difficult for two reasons. First, electrical networks require reliable, constant power delivery to consumers, whereas variable sources deliver power in a stochastic manner. Many renewable sources (wind farms, solar power) may ramp up or down unpredictably and faster than can be compensated for by the traditional power grid without idling of traditional generator resources. Towards this end, distributed storage is proposed as a solution for maintaining constant power delivery; however, massive and economical power storage is not yet readily available. Distributed generation has been introduced in traditional grids in the past with an important ancillary role in ensuring backup power in case of

malfunction, and in supporting the variations in loads due to consumer demand dynamics.

Secondly, variable generation introduces the problem of bi-directional flows in the grid, because customers are able to become occasional producers rather than pure consumers, and can send energy to the grid using the same infrastructure that has been engineered to be uni-directional. Thus the grid must be able to handle both incoming and outgoing flows from distributed storage. Current infrastructures are designed for uni-directional power flow. Under certain conditions renewable sources can cause a reversal of energy flow (Figure 2) and trigger protection mechanisms that may disable portions of the grid.

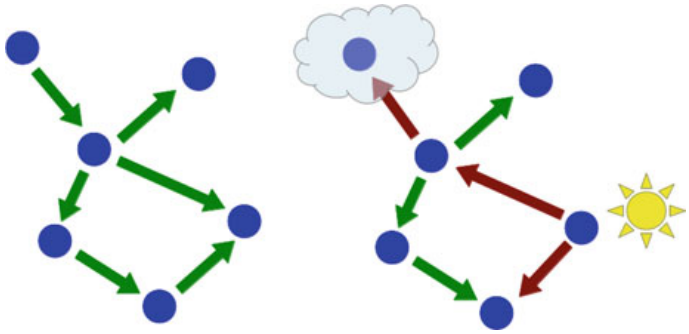


Figure 2: Reversal of energy flow in the presence of renewable energy sources. *Left:* A hypothetical power flow in a distribution grid, according to the way in which the grid has been engineered. *Right:* A weather instability switches the direction of the power flows, eventually causing automatic protections to trip the lines. Reproduced from (SCC+14).

The optimal deployment of energy reserves to maintain reliability and meet operational requirements, while taking into account the uncertainty and variability of renewable energy resources, remains an open question (FP14).

Electric Vehicles: Grid-to-Vehicle and Vehicle-to-Grid

Another related challenge in smart grids is that of electric vehicles. Electric vehicles can both draw power from the grid at variable locations and

times (Grid-to-Vehicle, or G2V), and can also be used as generators and storage units to supply the grid (Vehicle-to-Grid, or V2G).

In the G2V case, vehicle charging will lead to a significant new load on the existing distribution grids. In V2G, the challenge is the availability of electric vehicles, since an electric vehicle can only deliver power to the grid when it is parked and connected to the grid (FMXY12). The latter case also opens the possibility for creating new energy markets from electric vehicle users selling spare energy on the grid. Ultimately the effect of electric vehicles on the grid is driven by human behavior, and should be considered in future models.

Demand response and smart metering

Demand response is the ability of users to dynamically change their electricity consumption in response to price signals, which reflect total user demand. For example, smart appliances might be inactive during periods of peak demand when energy prices are higher and instead consume when the price is lower, thereby also reducing total peak demand (Figure 3). Dynamic pricing and distributed generation are expected to significantly reduce electricity costs for consumers, (KBD+12), as well as for providers who will have to build less generating plants (LDS10). Demand management will require the development of several technologies, including smart metering to report energy consumption patterns and appliances that can adjust their behaviour according to the price index of electricity. Consumers can also tailor their preference for renewable energy sources.

1.3.2 Problems of resilience

The U.S. Department of Energy (DOE) defined "reliability" as "the ability of power system components to deliver electricity to all points of consumption, in the quantity and with the quality demanded by the customer" (OK01). The DOE also defined "resiliency" as "the ability of an energy facility to recover quickly from damage to any of its components or to any of the external systems on which it depends" (HBFD+10).

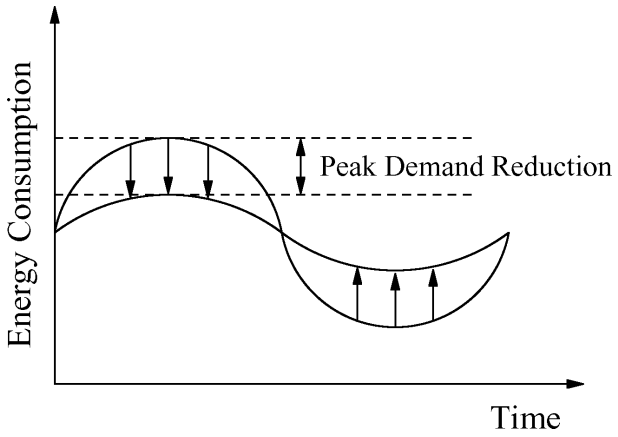


Figure 3: Peak demand reduction. Reproduced from (KBD+12) (© 2012 IEEE).

"Self-healing" is the process of dynamically and automatically restoring service in the system, and reacting quickly to disturbances in order to minimize their impact (LDS10). Traditional power grids mostly do not have self-healing capabilities. The failure of one link may result in loss of service to consumers until mechanical repair takes place. However, smart grids will have systems to monitor the status the of entire power grid and the means to adapt to disturbances.

The ability to self-heal in the event of failure is expected to be an important characteristic of smart grids (GWP+14). The future smart grid is envisioned to be generally adaptable and to have a reconfigurable network topology, enabling demand response and self-repair in the case of failures. The presence of redundant links is common in infrastructural networks, making self-healing feasible with current technology. Urban low-voltage distribution networks typically have a few inactive redundant links that can be activated to restore connectivity can be selectively activated to restore connectivity after a fault. These redundant links exist in the form of tie switches between different feeders, which can be activated or deactivated by opening or closing the switch. The future smart grid will have to leverage such resources in order to provide re-

silience and optimal power routing throughout the network. Similarly, by selectively activating or deactivating links, the future smart grid can optimally connect customers to energy sources in conformance to their preferences for certain types of energy sources and in order to minimize costs.

The smart grid is formed from the merging of a power delivery infrastructure and an ICT infrastructure (controllers, monitors, communication lines, etc.), as well as other entities such as generators, appliances, and their ancillary IT components that enable automation. Thus, the future smart grid will be an Internet of Things. The coupling and interaction between the power grid and other critical infrastructures such as the information and communication layer makes the smart grid a "multiplex" network, or a network of networks (Figure 4). This fusion however also carries risks for the vulnerability of the system. Failures in one network can trigger failures in the other, and propagate in cascades across the networks that can result in the disintegration of either (or both) of the networks (RP14). It is therefore important to investigate what kind of network topology and interconnection model maximizes the resilience of the smart grid.

1.4 Network Planning, and Fault Location, Isolation and Supply Restoration (FLISR)

The text in this section, excepting the last paragraph, was first published in (DSV17). Network Planning, and Fault Location, Isolation and Supply Restoration (FLISR) are both important functions of power distribution automation systems. In Power Systems, Network Planning broadly refers to the configuration at all levels of the network model: optimal grid construction, economical power distribution, forecasting of power demand, and monitoring the status of the distribution network (HRW16). In contrast with traditional power distribution networks, smart grids will allow much greater flexibility in their planning due to their reconfigurability (SXY$^+$16). This extends also to the structure of the grid's topology, which can be modified by the selective activation of lines and

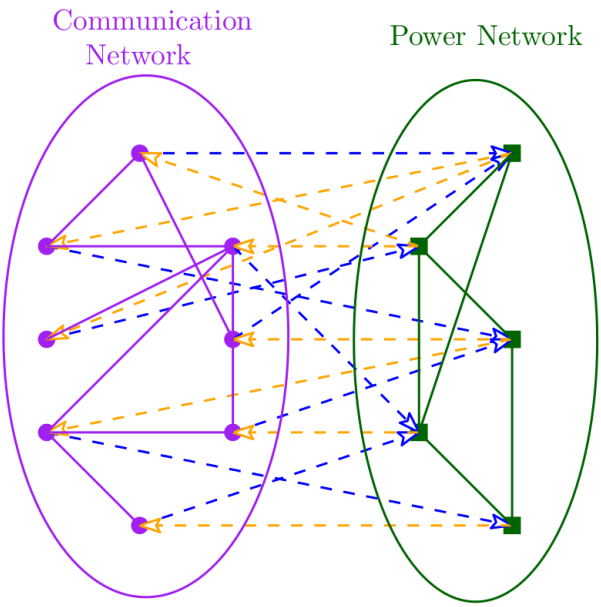


Figure 4: The smart grid as an interdependent complex network. Reproduced from (RP14) (© 2014 IEEE).

switches in the network.

FLISR aims at dealing with faults on power distribution lines, such as link failures that lead to power outages at customers' sites. The FLISR function consists of three steps: first the location of the fault has to be established, then the affected area has to be isolated by tripping the corresponding switches to protect the distribution lines and equipment, and finally the isolated customers need to get their power supply restored. The restoration is implemented by activating remotely operated switches (ROSes), which are located at the redundant tie switch sections connecting distribution feeders with each other.

An example of a FLISR problem formulation and solution is presented in (ZVD15) (© 2015 IEEE) and illustrated in Fig. 5. In the example in Fig. 5, there are three 11kV feeders supplied by three different zone substations (labeled A, B, and C). Distribution substations are po-

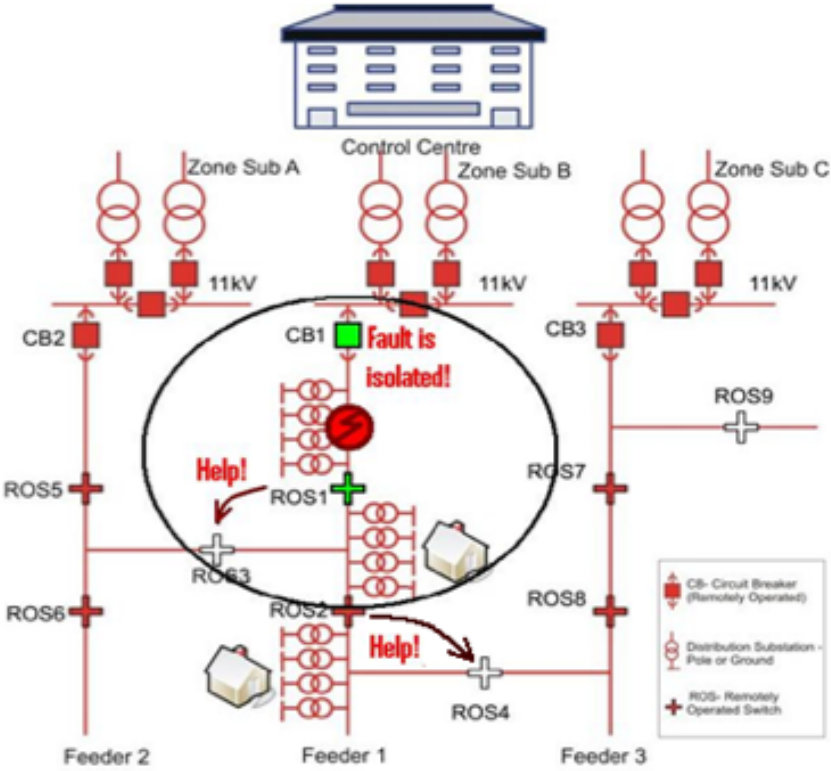


Figure 5: Sample distribution network, fault events and corrective actions. Reproduced from (ZVD15) (© 2015 IEEE).

sitioned along each feeder as demanded by the customers' loads. In the initial state, the switches ROS3, ROS4 and ROS9 are open, meaning that flow is disconnected. All other switches are closed.

An example FLISR scenario is described in (ZVD15) as follows: "A tree falls on the 11kV mains, severing the line between circuit breaker CB1 and remotely operated switch ROS1 at the location indicated and causing a permanent fault on feeder 1. The feeder protection trips circuit breaker CB1 at zone substation B. After an attempted automatic reclosure, CB1 locks out, preventing an accidental startup. This isolates the

fault but leaves customers on feeder 1 without power. As they are no longer energized, switches ROS1 and ROS2 propagate a "call for help" towards the substations of adjacent feeders 2 and 3. CB2 at zone substation A, and CB3 at zone substation C, respond with information about the headroom (excess capacity) available. This information propagates back down feeders 2 and 3. Switches ROS3 and ROS4 compare the available excess capacities with their respective loads. The switches agree on the steps necessary to restore supply: The mid-section of feeder 1 will be transferred to feeder 2; the tail section will be transferred to feeder 3; the head section will have to await repair. In the meantime, the control centre sends the crew to repair the located fault."

In this research we address the problem of Network Planning and FLISR in electrical distribution networks, through the automated configuration and re-configuration of the network's topology to optimally assign customers to energy sources. This is achieved via several heuristic algorithms that solve the combinatorial optimization problem of constructing a spanning forest with capacity constraints on each tree. Our ultimate aim is to also address the problem of self-healing, as demonstrated in the sample FLISR scenario in Figure 5. We define self-healing as the restoration of connectivity following a link or node failure that disconnects certain customers from the network. We achieve this by exploiting the redundant "dormant" links already present in the distribution network after the planning phase, activating them as needed to reconnect those nodes to the network.

Chapter 2

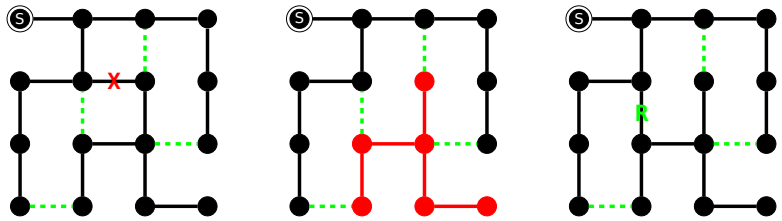
Problem Definition

2.1 Background

This work extends the work and ideas introduced in (QCS14). The central idea behind that work was to explore the resilience properties of different classes of networks by creating perturbations in the network and then restoring connectivity using a "self-healing" algorithm. The networks examined represent infrastructural networks. A failure event occurring at some location in the network causes a disconnection in the network, whereupon the self-healing algorithm activates unused but available redundant links to restore connectivity and functionality.

In the model in (QCS14) there is a graph representing the medium and low-voltage electrical infrastructure where nodes represent substations and edges represent arcs (power lines). A source node s provides power to the rest of the network, and other nodes are connected to the source by a spanning tree in order to be considered active. The graph contains redundant "dormant" links that are outside the spanning tree and are only used in the case of failures in order to re-establish connectivity. In the simulations in the work, failures can occur either in the links or on the nodes themselves, which has the effect of disabling all the outbound links of the node.

The authors developed a healing protocol based on a Minimum Span-



(a) Initial **(b)** Failure **(c)** Recovery

Figure 6: 6a) A source node (upper left corner, marked "S") serves 15 customer nodes connected to it in a tree-like distribution network. Dashed green lines represent dormant backup links that can be activated in case of failures. The link marked **X** is about to fail. 6b) A link fails, disconnecting six nodes (in red) from the source. 6c) A dormant backup link (marked **R**) is activated, restoring supply to all nodes.

ning Tree algorithm that reconstructs the maximal spanning tree that reconnects as many nodes as possible to the source. Figure 6 shows the operation of this protocol.

The number of uncorrelated failures in the network were varied and the performance of the self-healing algorithm on different classes of networks was assessed through experimental simulation. The results showed that the planar (square grid and tree-like) networks most similar to real-world electrical infrastructures are the most fragile, while scale-free (power-law degree-distributed) and small-world (having "shortcut" links connecting distant nodes) networks are much more resilient as the number of failures increases.

The model in (QCS14) considered networks having only one source. Thus, an open problem is to extend the model to consider also networks served by multiple sources. The sources can represent multiple generating stations, including different forms of electricity providers. Among these, for example, could be distributed "green" generators from renewable sources, whose energy production is volatile. This type of generation is becoming an increasingly relevant if not fundamental aspect of

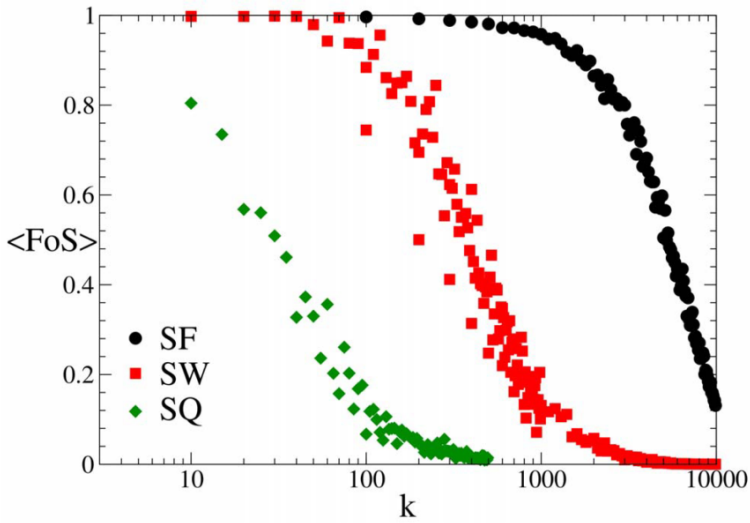


Figure 7: Comparison of resilience in different network structures. $< FoS >$ is the fraction of nodes receiving power. k is the number of failures introduced. SF: Scale-free. SW: Small-world. SQ: Square grid. Reproduced from (QCS14).

the future envisioned smart grid, as a shift is anticipated from the traditional hierarchical, unidirectional and capillary transport and distribution infrastructure to one that will support local energy trading among prosumers (PA13). In addition, distributed generators are of interest to studies of resilience of power grids as their limited presence in power grids has been found to increase resilience (SMC+13).

2.2 Model description

The text in this section was first published in (DSV17). This work extends research by (QCS14) in which smart grids were modeled as graphs and the resilience properties of various types of graphs were tested. The previous work explored network connectivity on cases with a single energy source, and left open the problem of cases featuring multiple energy

sources, which more closely resembles resembles the smart grid energy market and real-life scenarios.

Following the model in (QCS14), we represent the electrical distribution grid as an undirected graph in which nodes represent energy sources, customers, or other operational elements (switches, circuit breakers and substations), while edges represent arcs in the power network. Each source is assigned a capacity indicating the amount of commodity (electric power) that it can provide to the network, while each customer node has a demand in energy units. We refer to this as the *underlying graph* of the network. In the Network Planning phase, our objective is to connect each customer to a source, such that the capacity of the source is not exceeded by the combined demand of the customers it supplies. We do this by selecting edges in the underlying graph to construct a spanning tree from each source to the customers it will serve. This tree topology mimics real-life distribution networks, which are planar and radial (tree-like), and designed as such in order to optimize economic costs and facilitate management tasks such as metering consumption by any single node. We refer to this tree formed by the source, its customers, and the set of edges connecting them as the *active tree*. Any unused edges from the underlying graph that are not in any active tree are dormant links that can be activated as needed, either at the planning stage or in the case of failure.

On a graph representing an energy network with multiple sources, the problem of connecting every customer to a unique source becomes that of constructing a *spanning forest* (see Fig. 8) consisting of several trees rooted at their respective sources, with each tree sharing no customers with the others; that is, the trees must be vertex-disjoint. The union of all trees should then cover the entire set of customers. This is, in other words, the problem of creating a spanning forest with a capacity constraint on each tree bounding its total demand weight or cardinality. Each tree of the forest corresponds to a set of customers, rooted at a certain source.

The trees are vertex-disjoint so that no customer may be served by more than one source. As in the single-source case, in the event of a

disruption inactive links can be activated to recover connectivity.

The trees must be carefully constructed because each source has a capacity limiting the number of customers it can supply, and trees cannot share any vertices or edges (so they cannot be built arbitrarily), yet we wish to maximize the number of nodes in the forest. Thus, finding a spanning forest with these constraints is a discrete combinatorial optimization problem, and, as we will demonstrate in Chapter 4, one that is NP-complete.

We call this problem the CAPACITATED SPANNING FOREST problem (CSF), and as we will show, CSF is NP-complete even on unweighted graphs with two sources. To the best of our knowledge, this problem is hitherto not addressed directly in the literature. While the single-source case can be solved using a simple Minimum Spanning Tree algorithm, we shall show that cases with multiple sources are NP-complete.

The sources in a graph represented in this way can represent either constant or intermittent sources; for the representation of intermittent sources, an intermittent source is either added or removed from the graph (depending on its current state) and the solution to the graph is recalculated. For now, flows are not considered in the problem being addressed.

Model implementation

To implement the model described we used the NetworkX package for Python (HSS08). For the development of an algorithm that solves CSF we used a simulation-based approach to investigate certain heuristics, as outlined in later chapters. The evaluation metric used is the fraction of customers served $\langle \text{FoS} \rangle$.

2.3 Mathematical Problem Description

The CAPACITATED SPANNING FOREST problem (CSF) is the problem of covering the vertices of a graph with a spanning forest consisting of k disjoint trees rooted at different source nodes, with each tree having a capacity bounding the sum of the weights of its vertices.

In terms of smart grids, the graph is a representation of a low or medium-voltage distribution network. The non-source vertices in the graph correspond to customers and their demands are represented by the vertex weights. Sources represent sources of electrical energy to be distributed on the graph, with their capacities representing the capacity of energy they can serve.

2.3.1 Problem statement

The general CAPACITATED SPANNING FOREST (CSF) problem is as follows: We are given a finite simple graph $G = (V,\ E)$ with vertex weights $w_v \in \mathbb{R}^+$ for all vertices $v \in V$, a set of k source nodes $\{s_1, \ldots, s_k\} \subseteq V$, and a set of capacities $\{c_1, \ldots, c_k\} \subset \mathbb{R}^+$ that correspond to each source, i.e. source s_i has capacity c_i for $i = 1, \ldots, k$.

The objective is to cover the maximum possible number of vertices in G using k vertex-disjoint trees (i.e. trees sharing no vertices with another tree), such that each tree T_i is rooted at a different source s_i (for $i = 1, \ldots, k$), and the sum of the weights of all vertices in T_i is at most c_i.

An example is shown in Figure 8.

More precisely, if $F = \{T_1, \ldots, T_k\}$ is a spanning forest on G of k disjoint trees, the objective function of the optimization problem is $\max \sum_{v \in F} w_v$, subject to the constraints that have been mentioned above:

1. Each tree T_i must be rooted at a single source node s_i.
   $s_i \in T_i,\ (\{s_1, \ldots, s_k\} \setminus s_i) \cap T_i = \emptyset,\ i = 1, \ldots, k$

2. Each tree T_i is disjoint from the other trees in F.
   $\forall (T_i, T_j) \in F \times F,\ i \neq j\ :\ V(T_i) \cap V(T_j) = \emptyset$

3. The total weight of the vertices of T_i must not exceed its capacity c_i.
   $\sum_{v \in T_i} w_v \leq c_i,\ i = 1, \ldots, k$

This problem is NP-complete, as is demonstrated in the proofs in Chapter 4.

Note that vertex-disjointness implies also edge-disjointness. In practical scenarios, $n \gg k$, where $n = |V \setminus R|$, where R denotes the set of

source nodes. The weight of the sources is set to 0. Although the weights and capacities can be any real numbers, in our experimental scenarios we typically assign $w_v = 1$ for all non-source nodes, and all capacities are positive integers. However, the proof of hardness and the algorithms we present in subsequent chapters may easily be extended to the cases where w_v are heterogeneous and $w_v, c_s \in \mathbb{R}^+$.

The sum of the capacities of the sources can sum up to any value, as the objective function of the problem is to maximize the total number of vertices covered in the graph with the given capacities. However, in practical scenarios the capacities should sum up to at least the total number of non-source vertices. In the classic variant of the problem, we assume that the capacities of the sources equal the number of non-source vertices in the graph.

In a graph with n vertices and k source nodes, a spanning forest F will have exactly $n - k - 1$ edges.

The decision (or recognition) version of the problem can be stated as follows:

CAPACITATED SPANNING FOREST

INSTANCE: A graph $G = (V, E)$, a set of vertex weights $\{w_1, \ldots, w_{|V|}\} \subset \mathbb{R}^+$, a set of k source nodes $\{s_1, \ldots, s_k\} \subset V$, a set of capacities $\{c_1, \ldots, c_k\} \subset \mathbb{R}^+$.

QUESTION: Is there a vertex-disjoint spanning forest F on G consisting of k vertex-disjoint trees, such that each tree $T_i \in F$ is rooted at a distinct source s_i, and the sum of vertex weights in each tree does not exceed the capacity of its source c_i?

This formulation is useful for proving NP-completeness, since the class of NP-complete problems is composed of decision problems, i.e. yes or no questions.

2.4 Integer Program formulation

The CAPACITATED SPANNING FOREST problem can be expressed in the following Integer Program (IP) formulations, one vertex-centric and one edge-centric. The edge-centric program uses a $k \times m$ matrix (where k is the number of sources or trees in G and m is the number of edges in G of

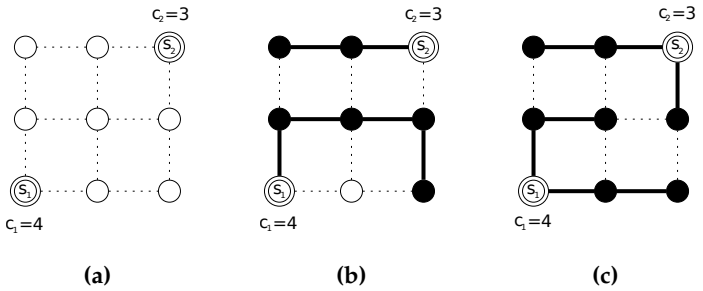


Figure 8: The CSF problem on a 9-node instance with two sources s_1 and s_2, and source capacities $c_1 = 4$ and $c_2 = 3$. Underlying graph in 8a. A sub-optimal solution in 8b, where a node (white) is left uncovered. An optimal solution in 8c, in which all nodes are covered.

variables x_{ie} for every edge in G and for every tree T_i, which can take the value 1 if the edge was selected for inclusion in the capacitated spanning forest F or 0 if it was not selected. The vertex-centric program uses a $k \times n$ matrix of x_{iv} variables for every vertex, where n is the number of vertices in G.

In addition to the constraints mentioned in Section 2.3.1, the following are the constraints to be satisfied in the program:

Constraints

1. Cardinality: $|T_i| \leq c_i,\ i = 1, \ldots, k$

2. Acyclicity, a.k.a. the "subtour elimination" constraint

3. Vertex-disjointness: Let $F = \{T_1, \ldots, T_k\}$
   $\forall (T_i, T_j) \in F \times F,\ i \neq j\ :\ V(T_i) \cap V(T_j) = \emptyset$

4. Fixed root locations: $s_i \in V(T_i),\ i = 1, \ldots, k$

5. Connectedness: $\forall T_i \in F, |E(T_i)| = |T_i| - 1$

Integer Program (edge-centric)

First, replace each undirected edge (u, v) in G with two directed edges $u \to v$ with weight w_v and $v \to u$ with weight w_u. Let m be the number of edges in this graph.

maximize $\displaystyle\sum_{i=1}^{k} \sum_{e \in E} x_{ie}$

subject to

1. $\displaystyle\sum_{e \in E(T_i)} w_e x_{ie} \leq c_i, \qquad i = 1, \ldots, k,$

2. $\displaystyle\sum_{e=(u,v) \in E : u \in S, v \in S} x_{ie} \leq |S| - 1, \qquad \forall S \subseteq V(T_i),\ i = 1, \ldots, k,$

3. $\displaystyle\sum_{i : v \in V(T_i)} x_{ie} \leq 1, \qquad \forall e = (u, v) \in E,$

4. $\displaystyle\sum_{e=(s_i,v) \in E(T_i)} x_{ie} \geq 1, \qquad \text{if } |E(T_i)| \geq 1,\ i = 1, \ldots, k,$

5. $\displaystyle\sum_{e \in E(T_i)} x_{ie} = |V(T_i)| - 1, \qquad i = 1, \ldots, k,$

6. $x_{ie} = \{1, 0\} \qquad i = 1, \ldots, k,\ e = 1, \ldots, m.$

Integer Program (vertex-centric)

$$\text{maximize} \quad \sum_{i=1}^{k} \sum_{v \in V} x_{iv}$$

subject to

1. $$\sum_{v \in V(T_i)} w_v x_{iv} \leq c_i, \qquad i = 1, \ldots, k,$$
2. $$\sum_{v \in V(S)} x_{iv} \geq |S| + 1, \qquad \forall S \subseteq E(T_i),\ i = 1, \ldots, k,$$
3. $$\sum_{i=1}^{k} x_{iv} \leq 1, \qquad \forall v \in V,$$
4. $$x_{iv} = 1, \qquad i = 1, \ldots, k,\ v = s_i,$$
5. $$\sum_{v \in V(T_i)} x_{iv} = |E(T_i)| + 1, \qquad i = 1, \ldots, k,$$
6. $$x_{iv} = \{1, 0\}, \qquad i = 1, \ldots, k,\ v = 1, \ldots, n.$$

For the Linear Programming relaxation of the above Integer Programs, $0 \leq x_{ie} \leq 1$ and $0 \leq x_{iv} \leq 1$.

In general, integer programs cannot be solved in polynomial time, while their linear program relaxations (as well as linear programs in general) are polynomial-time solvable. However, the linear programming relaxation of an integer program can only give an approximation of the integer program solution, namely a lower bound in the case of minimization problems or an upper bound in the case of maximization problems. Furthermore, because there is an exponential number of the subtour elimination constraints (marked 2) with respect to the input size for both of the integer programs presented, solving these programs using standard methods such as the simplex method is impractical. Therefore we must use a technique known as the ellipsoid method that can solve them in polynomial time, which entails defining a separation oracle for finding violated constraints. However, as mentioned, even the solvable linear program provides only an approximation of the optimum of the integral problem; thus either it must be used in combination with heuristics such as Branch-and-Bound techniques, or other approaches such as

approximation algorithms and other heuristic algorithms can be pursued. In this work, we focus on heuristic algorithms and local search rather than techniques involving mathematical programming.

2.5 Application and Motivation

As mentioned earlier, the primary motivation for this research is an application in smart grids. In the smart grid context, the tree roots represent energy sources (powerplants, renewable generators, batteries, etc.) and their capacities represent the quantity of energy that they can distribute. Other vertices in the graph represent customers, while edges represent power lines. Customers are connected to a power source by activating an edge between themselves and a tree routing back to a source. The customer node is then added to the tree. In the weighted case, the weights on each vertex represent the power demand of a given customer node. By introducing directed edges on the graph, uni-directional flow can be represented.

Another possible application of this problem is towards military applications. Consider a graph where nodes represent strategic objectives such as towns or other locations of strategic value and edges represent the transport links between them. The trees of the CSF problem can represent different military detachments that start from insertion points (sources) and expand to cover as many objectives as possible. The capacities of the sources represent the resources an army group has to achieve its goals, such as fuel or personnel, while node weights represent the resource cost to secure an objective. The trees must stay connected to their sources so that supply lines to all objectives remain undisrupted.

Lastly, another application is the case of computer networks, where a set of service providers with differing resources must serve a set of clients, and providers do not have direct links to all clients.

Chapter 3

Previous work and review of existing literature

To date, there is no previously published work on solving the general CSF problem as it is defined in the problem description in Section 2.3.1, nor for special cases of interest such as unweighted grid graphs with integral capacities. To the best of our knowledge, our works in (DSV17) and (DV17) are the first to present algorithms for CSF. Although to the best of our knowledge CSF is not reported in the literature, a few similar problems are known and have been studied. Some of these problems are very similar to CSF, while others may be in a related family of problems. These other problems, however, are not identical to CSF. This means that although they may inform our approaches toward developing solutions for CSF, the results and algorithms that emerged from the study of those problems are not directly applicable to CSF. We will describe some similar problems in the literature with the relevant reference to their proof of NP-completeness while noting their differences from CSF and demonstrating why new approaches must be developed for tackling this problem.

In Computer Science, often problems that appear similar to one another are not mutually interconvertible, meaning that small differences in the objective function or constraints result in distinct problems that

could even belong to different complexity classes, and approaches to solving one problem are not guaranteed to work for another. A simple example of this are the shortest path problem and the longest path problem, which, although superficially appearing similar, in fact belong to different complexity classes as shortest path can be solved in polynomial time while longest path is NP-complete. Although the Bellman-Ford algorithm for the shortest path problem can be adapted to compute the longest path by negating all edge weights, this is only true for Directed Acyclic Graphs and not graphs in general. These differences also exist across problems from the same complexity class: for example, approximation algorithms designed for the NP-complete Traveling Salesman Problem are specific to TSP and cannot be directly applied, for example, to the Vertex Cover problem. Thus, it is often the case that results from one problem are not transferable to another problem, except in special cases of the problem.

3.1 Problems in the literature similar to CAPACITATED SPANNING FOREST

CAPACITATED MINIMUM FOREST (LSX13)

The most similar problem to CAPACITATED SPANNING FOREST in the literature is CAPACITATED MINIMUM FOREST (CMF) introduced by Liang et al. (LSX13), a problem motivated by an application in wireless sensor networks. The objective in this problem is to find a minimum-edge-cost spanning forest on an edge-weighted Euclidean graph given a set of roots, with upper bounds on the cardinality of each component. Liang et al. prove this problem to be NP-complete by reduction from CAPACITATED MINIMUM SPANNING TREE (Pap78a) and provide approximation algorithms for it. The decision version of the problem is formulated as follows:

CAPACITATED MINIMUM FOREST
INSTANCE: A finite graph in the Euclidean plane $G = (U \cup R, E)$, where R

is a set of roots and U is a set of non-root nodes. $E = (U \times U) \cup (R \times U)$, and for every edge $(u, v) \in E$ there is a cost w_e equal to the Euclidean distance between u and v. All roots have capacities given by the function $c : R \to \mathbb{N}$, such that $\sum_{s \in R} c(s) = |U|$. Finally, an integer K is given.

QUESTION: Is there a spanning forest F of cost at most K for which every connected component contains exactly one root s_i $(i = 1, \ldots, |R|)$ and exactly $c(s_i)$ other vertices?

Note that in the optimization version of the problem, the objective is to minimize $\sum_{e \in F} w_e$, subject to the constraints above.

Liang et al. provide approximation algorithms for three variations of the problem that deliver an approximation ratio of at most 2 and run in $O(n^3)$ time. Their approach derives from a general approximation technique for constrained forest problems introduced by Goemans and Williamson (GW95). The authors state that this is the first such algorithm in the literature for this problem. More recently, Jaiswal et al. provide a similar approximation algorithm for the same problem (Jai15) (JS15).

CMF however differs significantly from CSF in the following ways:

1. An instance of CMF has weighted edges and unweighted nodes, while in CSF vertices are weighted and edges are unweighted. This leads to two essential differences between the problems. The first is that the capacity constraint on the trees in CMF is essentially a cardinality constraint, affecting only the number of nodes in a given tree. This is in contrast with CMF, in which vertices can have heterogeneous demands, which must in total not exceed the capacity of the tree. Second, because edges are weighted in CMF instances, the objective of CMF is to find a spanning forest of minimal cost, i.e. minimize $\sum_{e \in F} w_e$. This objective function does not exist in CSF.

2. The input graph is a nearly complete graph in which, letting R denote the set of all roots, $E = (U \times U) \cup (R \times U)$, while in CSF the graph is arbitrary.

3. The cardinality of a tree $T_i \in F$ in CMF must be exactly $c_i + 1$ and

must be a natural number, whereas in CSF it can be less than c_i and can be a real number.

4. $\sum_{i=1}^{|R|} c(s_i)$ must necessarily equal $|U|$ in CMF. In fact, the algorithms suggested by Liang et al. terminate if this condition cannot be met. This is possible because CMF graphs are nearly complete and have Euclidean weights weights, which is not the case in CSF.

The special case of CSF where $w_u = 1 \; \forall u \in U$ could be framed as a CMF problem of minimizing the edge weights of a spanning forest on a graph with uniform edge weights (e.g. $w_e = 1 \; \forall e \in E$). However, this would not be a valid instance of CMF because of the requirement that graphs be Euclidean, meaning that a CMF input graph cannot have uniform edge weights. Further, the case of uniform unit vertex weights would represent only a special case of CSF.

We therefore believe these differences to be sufficient to distinguish CSF from CMF, and to necessitate a different algorithmic approach than the ones proposed in (LSX13).

GRAPH TREE PARTITION PROBLEM (CM04)

Cordone and Maffioli (CM04) discuss the complexity of the Graph Tree Partition Problem (GTPP), a general family of problems that can be variously defined through combinations of three constraints: root, inclusion, and weight, and four objective functions: max-sum, min-sum, max-min, and min-max. The basic problem is: given an edge-weighted graph $G = (V, E)$, partition the vertex set V into k disjoint vertex subsets U_i and build a spanning tree T_i on each of them, subject to any of the aforementioned constraints. The objective function can be either maximizing or minimizing the sum of the edge weights in the forest (max-sum, min-sum), or minimizing the total edge cost of the most expensive tree in the forest (min-max), and its reverse, maximizing the cost of the cheapest tree (max-min).

This problem is a generalization of many problems involving partitioning a graph into trees. The authors describe three variations of constraints for this problem: root constraints, inclusion constraints and

weight constraints. Of relevance to CSF are the root and weight-constrained problems. Root constraints require that each tree T_i contains at least one of the vertices in a given root set R_i, i.e. $T_i \cap R_i \neq \emptyset$. Weight constraints describe constraints on the total weight of the vertices of the tree, and are defined similarly as in CSF, though over an interval, and must be natural numbers. Given a weight function $w : V \to \mathbb{N}$ on the vertices of the graph, the total weight of each tree $w(T_i) = \sum_{v \in T_i} w_v$ must belong to a given interval $[W_i^-; W_i^+], W_i^- \leq w(T_i) \leq W_i^+$.

Cordone and Maffioli show that the weight-constrained problem is NP-complete for all four objective functions, irrespective of any additional constraints. The root-constrained problem is NP-complete only for min-max and max-min problems. Previously, (GBH97) had shown that the weight-constrained problem is strongly NP-hard for the case where $w_v = 1 \; \forall v \in V$, and $W_i^- = W_i^+ = n/k, i = (1, \ldots, k)$, i.e. vertices are unweighted and all trees must have the same cardinality. This is the MINIMUM TREE PARTITION problem discussed in (GBH97).

The root-constrained min-max GTPP is a generalization of the Mini-Max Spanning Forest Problem (MMSFP) discussed by Yamada et al. (YTK96), in which the root sets R_i are singletons. This special case of the GTPP was already proven NP-complete by (YTK96), who provided a branch and bound algorithm for it.

Anticipating the proof on Chapter 4, a reduction from the GTPP can be used to prove that CSF is NP-complete.

Theorem 1. *GTPP $\leq_P$ CSF for GTPP instances with weight and root constraints, uniform edge costs and singleton root sets $|R_i| = 1 \forall i$.*

Proof. Given any weight-constrained and root-constrained instance of GTPP B with uniform edge costs and singleton root sets, it is possible to build an instance of CSF A such that a solution to the former exists if and only if there exists a capacitated spanning forest on A.

To do so, we first observe that any weight-constrained and root-constrained instance of GTPP is hard even in the case of uniform edge weights $w_e = 1$ and singleton root sets $R_i = \{s_i\}$. This is can be demonstrated with the same gadget used in the proof by Cordone and Maffioli, in which all edges are of cost 1 with the exception of some edges of cost γ that are

unused except to complete the graph and show its validity under the triangle inequality. The proof still holds even if we remove the edges of cost γ and collapse the sets R_1 and R_2 to the single nodes r_1 and r_2, adjusting $W = n + nm + 1$ for each tree.

Lemma 1. *GTPP remains NP-complete for instances with uniform edge cost and single root nodes.*

By restricting our GTPP instance B to those with uniform edge weights and singleton root nodes, we construct a CSF instance A by preserving the same nodes, node weights, edges, roots and capacities in B, and ignoring the costs on the edges and total edge cost requirement K. Since all edge costs are uniform, a feasible solution to the GTPP will have cost $K = |E| - |R|$. A capacitated spanning forest on this graph will obey the edge cost, root and weight constraints from the GTPP instance B, since there will be K edges in the forest, the roots are the same as in GTPP and the trees have the same weight (or capacity) constraints. Thus a solution to CSF is a solution to GTPP and vice versa. □

We have shown that certain instances of CSF where $w_v, c_i \in \mathbb{N}$ can be considered special cases of GTTP where $w_e = 1$ for all edges. However, because of the different objective functions between the two problems, we consider it more cumbersome to express CSF in terms of maximizing the total cost of edge weights when there generally are no edge weights for CSF. Furthermore, the weight constraints as defined by Cordone and Maffioli must be natural numbers, while in CSF they can be any positive real numbers.

Some other key differences pertinent to the proof of NP-completeness are that Cordone and Maffioli's proof for the weight-constrained case uses a bound on the total edge cost and non-uniform edge weights, the weight constraints for both trees are the same, and the roots, although indicated, are not fixed. We will give a proof of the NP-completeness of CSF using no edge cost bound and only weight constraints, heterogeneous weight constraints, and fixed roots, which we believe is a stronger result for our problem.

While the authors analyze the complexity of Graph Tree Partition Problems, they do not give suggestions for how to attack the NP-complete

variations of GTPP. Nor are algorithms suggested for the rooted, weight-constrained special case with unweighted edges that resembles the CSF.

CAPACITATED MINIMUM SPANNING TREE (Pap78a)

A Capacitated Minimum Spanning Tree (CMST) is a minimal cost spanning tree of a graph with a designated root node s in which every subtree branching from s has a cardinality (defined as the total number of nodes in the subtree) no more than a capacity constraint c. The edge cost of the CMST may be no more than K. Each subtree is connected to the root by a single edge; thus, if s and its incident edges were removed, there would be a minimum-cost spanning forest of trees of cardinality at most c. This problem was introduced and demonstrated to be NP-complete by (Pap78a).

This problem differs from CSF in several ways:

1. The roots of the subtrees are not specified.
2. All subtrees have the same cardinality constraint c.
3. Vertices are not weighted.
4. In CMST there are edge costs and the objective is to minimize the total edge cost, which is not the case in CSF.

The objective of minimizing the total edge cost of the forest is essential to the definition of CMST, as the proof of NP-completeness in (Pap78a) requires edge costs in order to be valid, and does not hold when edge costs are uniform or ignored. In the proof of (Pap78a), the edge cost of the CMST may be no more than K. When this constraint is removed or edge costs are uniform, a solution to CMST does not necessarily correspond to a solution to SAT, so the reduction does not hold. Figure 9 demonstrates this. In Chapter 4 we will demonstrate that CSF is NP-complete even for cases with no edge costs, a stronger result than the one in (Pap78a).

The Esau-Williams heuristic and its variations are the most established approaches for CMST.

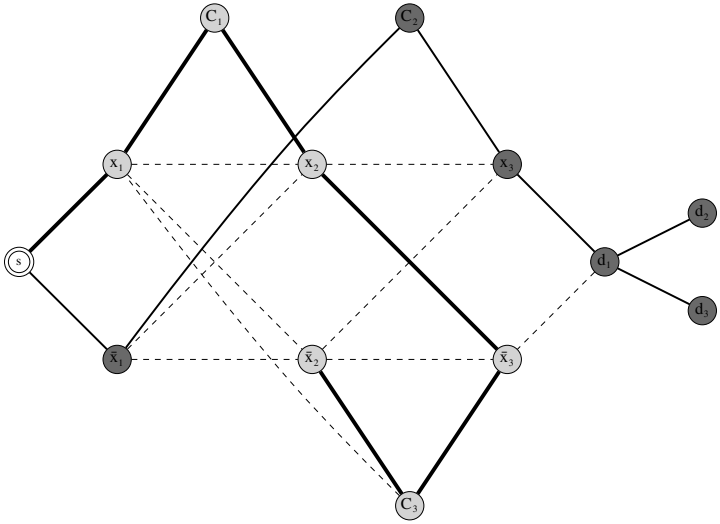


Figure 9: Demonstration that in a CMST instance with no edge weights, the solution to CMST is not necessarily a solution to SAT. Illustrated is the instance of CMST used in the proof of NP-completeness from (Pap78a) corresponding to the boolean satisfiability formula $\phi = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$, however with $w_e = 1 \; \forall e \in E$. Each subtree connected to source s has a capacity $c = 6$ and is denoted by solid bold lines of different thickness. However, although this is a valid solution for CMST without edge weights, the corresponding selection of literals does not satisfy ϕ. Indeed, it is even a logical contradiction as both x_2 and $\bar{x}_2$ are selected.

3.1.1 Other problems similar to CSF

Following is a list of NP-complete problems problems that are similar to CSF, but not as similar as the previously mentioned CMF, GTPP, and CMST. These problems all differ from CSF in either their objective function or their constraints, typically due to having weighted edges rather than vertices and an objective of minimizing the total edge cost, and lacking either the capacity or the root constraints. Because of these differences, CSF cannot be considered a special case of any of them. Furthermore, the proofs of hardness and algorithms analyzed for those problems can also thus not apply to CSF. These problems are often similar to each other, differing in only a constraint or objective function, and are sometimes reported in the literature without reference to the other problems.

BOUNDED COMPONENT SPANNING FOREST (GJ79)

Given a graph $G = (V, E)$ where each vertex has a non-negative weight, and two positive integers k and c, can we partition the vertices of V into k (or less) disjoint subsets where each subset is connected and the sum of the weights of the vertices in each subset is at most c?

While similar to CSF, BCSF (also known as [ND10] in (GJ79)) has the following differences from it:

1. There are no fixed roots specified for the trees. This means that trees can be placed anywhere on the graph, whereas CSF has the necessary constraint that a tree of a given capacity must include its given root.
2. The exact number of trees is not given, only a maximum number k.
3. All trees have the same capacity c.
4. Capacities and vertex weights are strictly integers.

A few results from this problem are relevant to CSF. Björklund et al. (BHK09) (BH06) presented exact algorithms for a whole class of partitioning problems including BCSF that run in $2^n n^{O(1)}$ time and exponential space. Their method is based on the inclusion–exclusion principle

and an algorithm known as the zeta transform. They also present algorithms running in $O(n^{O(1)})$ space and $3^n n^{O(1)}$ time if membership in 2^V can be decided in polynomial time, which is the case for BCSF.

It is known that certain NP-complete problems are decidable in polynomial or pseudo-polynomial time for certain classes of graphs that are of fixed bounded treewidth, i.e. having a treewidth of at most a constant t (Bod86). A graph's treewidth is defined as the minimum width among all possible tree decompositions of a graph, while a tree decomposition is a mapping of a graph onto a tree. BCSF is one such problem (ALS88) (Bod88). Furthermore, according to Courcelle's theorem, all graph properties that are definable in monadic second-order logic can be decided in linear time for classes of graphs of bounded treewidth (Cou90). Shortly after the publication of Courcelle's theorem, it was found that BCSF can also be expressed in this way as a language. (ALS91) . Unfortunately, the general planar graphs do not fall into this category, and neither does the $n \times n$ grid that has treewidth at most n. Additionally, deciding that the treewidth of a graph $G \leq t$ is NP-complete (ACP87). These findings suggest that CSF, which has a more restrictive graph topology, might also be NP-complete for the planar graph and grid cases that we are interested in.

MIN-SUM TREE PARTITION & MIN-MAX TREE PARTITION(GBH97)

Given a complete, edge-weighted, undirected graph $G = (V, E)$ together with edge weights that satisfy the triangle inequality, partition V into k equally-sized subsets and find a minimum spanning tree on each of the subgraphs induced by the partition. The objective of the min-sum version of the problem is to minimize the total sum of the edge costs of all the spanning trees in the forest, while in the min-max version it is to minimize the cost of the most expensive tree.

Stated more formally, given a graph $G = (V, E)$ with $|V| = n$, where $n \mod k = 0$, and edge weights satisfying the triangle inequality $w_e, \forall e \in E$, the MIN-SUM TREE PARTITION problem is to find a partition of V into k vertex-disjoint trees $\{T_i\}_{i=1}^{k}$, $|T_i| = n/k$, such that $\sum_{i=1}^{k} \sum_{e \in E(T_i)} w_e$

is minimized. For the MIN-MAX TREE PARTITION problem, minimize $\max_{1 \le i \le k} \{ \sum_{e \in E(T_i)} w_e \}$.

MIN-SUM TREE PARTITION and MIN-MAX TREE PARTITION (MMTP) are both special cases of the weight-constrained min-sum and min-max GTPP, with the weight constraints $w_v = 1 \; \forall v \in V$ and $W_i^- = W_i^+ = n/k \; (i = 1, \ldots, k)$, i.e. all trees have the same number of vertices. Guttmann-Beck and Hassin (GBH97)(GBH98) devised approximation algorithms for these problems.

These problems differ from CSF in that there are no root constraints, all trees have the same number of vertices, and in the case of MMTP the objective function is to find a forest in which the cost of the maximum-cost tree is minimized.

MINIMUM TREE PARTITION (GBH98)

This problem is the same as MIN-SUM TREE PARTITION, but the trees can be of different cardinalities.

Given a complete weighted undirected graph $G = (V, E)$ together with edge weights satisfying the triangle inequality, partition V into k subsets of given size and find a minimum spanning tree on each of the subgraphs induced by the partition, minimizing the total sum of the spanning trees. In other words, find the minimum-cost spanning forest of k disjoint trees on G.

More formally: Given a graph $G = (V, E)$ with $|V| = n$ and k positive integers $\{c_i\}_{i=1}^k$ such that $\sum_{i=1}^k c_i = n$, and edge weights $w_e, \forall e \in E$ satisfying the triangle inequality, the MINIMUM TREE PARTITION problem is to find a partition of V into vertex-disjoint trees $\{T_i\}_{i=1}^k$ such that $\forall i \in \{1, \ldots, k\} \; |T_i| = c_i$, and $\sum_{i=1}^{k} \sum_{e \in E(T_i)} w_e$ is minimized.

This problem is proven NP-complete in (GBH97) for $k = 2$. (GBH98) present a $(2k-1)$-approximation algorithm running in $O(k^2 4^k + n^2)$ time. An extension of the algorithm for unequally sized trees maintains the $(2k-1)$ approximation ratio but runs in exponential time.

This problem differs from CSF in that no roots are specified for the

trees $\{T_i\}_{i=1}^{k}$ of the graph, and the vertices are unweighted. Lastly, the objective is to minimize the total sum of the edge weights on the trees.

MINI-MAX SPANNING FOREST PROBLEM (YTK96)

The mini-max spanning forest problem (MMSFP) requires finding a spanning forest of an undirected graph that minimizes the maximum of the costs of constituent trees. This problem was proven NP-hard by (YTK96).

Let $G = (V, E)$ be an undirected graph with each edge e having a cost $w : E \to \mathbb{Z}$. Given a set of root vertices $R := \{s_1, s_2, \ldots, s_k\}$, a rooted spanning forest F is a spanning forest of G consisting of k disjoint trees $T_1, T_2, \ldots, T_k$ such that $s_i \in T_i$ $(i = 1, 2, \ldots, k)$. MMSFP is the problem of finding the R-rooted spanning forest $F*$ that minimizes $\max_{1 \le i \le k} \{ \sum_{e \in E(T_i)} w_e \}$.

This is the same problem as MIN-MAX TREE PARTITION but with root constraints, or as a min-max GTPP where the root sets R_i are singletons.

This problem differs from CSF in several ways. First, there are no vertex weight bounds on the trees. The objective of MMSFP is minimizing the edge weight of the largest tree, while covering all the vertices, rather than covering all the vertices while restricted by a capacity constraint. Lastly, the graph is edge-weighted rather than vertex-weighted.

The authors introduce a heuristic algorithm and a branch-and-bound technique for exact solutions for small instances of problem.

CONSTRAINED FOREST (CI93)

Given an undirected edge-weighted graph and a natural number m, the CONSTRAINED FOREST PROBLEM seeks a minimum-edge-weight spanning forest such that each of its trees spans at least m vertices. (CI93) demonstrate that the problem is NP-hard for $m \geq 4$ and provide a simple 2-approximate greedy heuristic that runs within the time needed to compute a minimum spanning tree. (GW95) provide a widely used general 2-approximation technique for Constrained Forest problems running in $O(n^2 \log n)$ time. (LM05), (LM06) and (LM11) provide a series of heuristics for this problem.

This problem differs from CSF in several ways. Aside from the objective of minimizing total edge weights, there is a minimum rather than a maximum bound on tree size. There are also no fixed roots for the trees.

BOUNDED TREE COVER(KS11)

Given an edge-weighted graph $G = (V, E)$ with edge weights $w : E \to \mathbb{N}^+$ and a bound K, find a tree cover with a minimum number of trees such that each tree has total edge weight at most K. A tree cover is a set of subtrees of G $\{T_1, T_2, \ldots, T_k\}$ such that $\bigcup_{i=1}^{k} V(T_i) = V$. (KS11) present a 2.5-approximation algorithm for this problem.

The differences from CSF are that there are no tree roots in this problem, the number of trees is unspecified, the bound is on total edge weights rather than vertex weights, and the objective function is different than in CSF. Further, the trees in a tree-cover are not necessarily edge-disjoint (thus may share vertices as well).

MIN-MAX ROOTED k-TREE COVER (EGK$^+$03b)

A min-max rooted k-tree cover is a cover of all the vertices of a graph using k or less trees that minimizes the edge cost of the most expensive tree, with each tree having a distinct given root. The trees in the cover may share nodes and edges.

Stated more formally, given an undirected graph $G = (V, E)$ with positive integral edge weights $w : E \to \mathbb{N}^+$, a set of roots $R \subset V$ and a positive integer k, a ROOTED k-TREE COVER of G is a collection of at most k trees $\mathcal{T} = \{T_i\}$ such that each tree has a distinct root in R and $\bigcup_{i=1}^{k} V(T_i) = V$. The min-max problem is to minimize $\max_{T_i \in \mathcal{T}} w(T_i)$, where $w(T_i) = \sum_{e \in E(T_i)} w(e)$. The trees in a tree cover may share nodes and edges. The root of T_j may be in T_i $(i \neq j)$, but their root nodes must be distinct.

The min-max problem was studied by (EGK$^+$03a) and shown to be NP-complete by reduction from BIN-PACK. (EGK$^+$03a) also describe a polynomial-time $(4 - \epsilon)$-approximation algorithm for the min-max ver-

sion of this problem (where ϵ is an adjustable parameter controlling the trade-off between accuracy and running time of the solution).

This problem differs from CSF in that trees are not vertex-disjoint or edge-disjoint, and the trees are not capacitated and can contain any number of nodes. Further, the objective is to minimize the cost of the edge weights in the most expensive tree.

k-BALANCED PARTITIONING (Fel11)

The k-BALANCED PARTITIONING problem is to partition the n vertices of a graph into k sets of size at most n/k each, while minimizing the cut size (i.e. the number of edges connecting vertices from different sets). (Fel11) has shown that this problem is hard even on grids and trees. This means that attempting to partition even regular graphs such as a grid will be hard, and precludes approaches such as first obtaining a minimum spanning tree of the graph and then making cuts to partition the tree into a forest.

This problem differs from CSF in that the objective function is different (minimizing the cut size), the partitioned sets are not trees, the sets have no roots, and the sets must be of roughly equal size.

PARTITION INTO FORESTS (GJ79)

INSTANCE: Graph $G = (V, E)$, positive integer $K \leq |V|$.
QUESTION: Can the vertices of G be partitioned into $k \leq K$ disjoint sets $V_1, V_2, \ldots, V_k$ such that, for $1 \leq i \leq k$, the subgraph induced by V_i contains no circuits?

This problem differs from CSF in that the solution to this problem is a set of forests, not just one forest. There are also no constraints on the size of the trees (or forest), and no root nodes specified for the trees.

This problem is known as [GT14] in (GJ79) and also reported as MINIMUM k-CAPACITATED TREE PARTITION. Goemans and Williamson's general scheme (GW95) can approximate this problem within $2 - 1/|V|$ (CKH95). (Bod88) showed that [GT14] can be solved in polynomial time for graphs with bounded treewidth, and graphs with bounded treewidth

and degree. (BPT92) provided a method for solving this problem in linear time on families of recursively constructed graphs (graphs composed of smaller members of the same family, such as trees). Their method entails expressing the problem in their predicate calculus, which automatically generates a recurrence relation usable in a dynamic programming solution. (BHK09) reported a technique for the general problem of partitioning an n-set N into subsets $\{S_1, \ldots, S_k\} \subset N$, based on the inclusion-exclusion principle zeta transform, that can solve these types of problems in 2^n, or exponential time. This technique consists of expressing the problem as an inclusion-exclusion formula over the subsets of N, and then to use memoization via an algorithm known as the fast zeta transform. The authors claim that their method is extensible also to [GT14].

k-MINIMUM SPANNING TREE (RSM$^+$96)

Given a graph with non-negative edge weights and a natural number k, find a tree of minimum cost spanning exactly k vertices. (RSM$^+$96) proves this problem is NP-complete even for planar graphs and provide a $2\sqrt{k}$ approximation algorithms for it. A sequence of improvements led to $(2 + \epsilon)$ approximation algorithms by (Gar05) and (AK05). Of note is that even the rooted version of this problem (i.e. where the solution must contain a given root node) is NP-hard (AK05).

SPANNING k-TREE FOREST(LZ09)

Given a graph, find a maximum spanning forest of k-trees, that is, a forest of maximum total edge weight where each component is a tree in which all vertices are at most distance k from the root of the tree. In the weighted edge distance version, k bounds the sum of edge weights in the shortest path between any root and its vertices. (LZ09) provide a polynomial time $\frac{k}{k+1}$-approximation algorithm for the general problem and a 0.5-approximation algorithm for the weighted edge distance version.

3.1.2 Other related problems

A class of problems very closely related to CSF is that of MULTIPLE-DEPOT VEHICLE ROUTING. This problem is nearly the same as CSF, except that instead of trees branching from the roots cycles are used to cover all nodes. It is known to be NP-hard.

As can be seen, CSF is not seen in the literature and approaches to other problems, while serving as a basis and inspiration, cannot be used directly for solving it. Therefore we must come up with our own.

A naive approach would be to take the MST of G and then iterate through the edges of the MST, making a cut. If the two components are of the size of the capacities and the roots are in the components, accept. The problem with this approach is that there is no guarantee for the existence of such components that also contain the roots. We would have to have the "right" Minimum Spanning Tree. The same problem holds for other naive approaches such as taking a path through G. This would not be a good approach to begin with, since the expected value of the number of nodes failing in case of a failure is higher for a path than for a tree, thus paths should be avoided in the design of resilient networks.

Chapter 4

The NP-completeness of CAPACITATED SPANNING FOREST

Summary

In this chapter we prove that the Capacitated Spanning Forest problem, introduced in Chapter 2 is NP-complete by reduction from the general boolean satisfiability problem (SAT). We show that CSF is NP-complete for cases with more than two sources and positive capacities. This NP-completeness result justifies the heuristic approaches taken by the authors in other works discussed in the later chapters of this thesis. This result was presented at (Dav18).

4.1 Introduction

CSF is NP-complete via a reduction from the general boolean satisfiability problem (SAT) introduced in (Coo71). In SAT, given a boolean formula ϕ that is a conjunction of m clauses over n boolean variables, we must find a satisfying truth assignment for the variables such that ϕ evaluates to 1 (or TRUE). For example $\phi = (x_1 \vee \bar{x}_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$

is a boolean formula in conjunctive normal form with three clauses and three variables (each symbol in a clause being called a literal), and a satisfying assignment $x_1 = 0, x_2 = 0, x_3 = 1$. We prove that SAT $\leq_P$ CSF, that is, SAT is polynomial-time reducible to CSF.

To do this we perform a reduction from SAT, the general boolean satisfiability problem, which is known to be NP-complete when at least one clause in the formula has more than two literals (Coo71) (GJ79). We proceed with the approach of a "reduction" as elaborated in (GJ79), namely a conversion of any instance of a known hard problem into a specific instance of our problem of interest. By reducing a known hard problem A to our problem B, we prove that B is at least as hard as A. The reasoning behind this is that if there was an efficient algorithm for solving B, there would also be an efficient algorithm solving A: simply first convert A to B, then use the algorithm for B on it to obtain a solution to A. However, this is a contradiction, as our assumption was that problem A is hard. Therefore, reducing a hard problem A to B proves that B is NP-hard, unless P=NP.

4.2 Proof of NP-completeness

4.2.1 Proof 1. CAPACITATED SPANNING FOREST is NP-complete for graphs with more than 2 sources.

Theorem 2. SAT $\leq_P$ CAPACITATED SPANNING FOREST *for graphs with* $|S| > 2$, *where* S *is the set of sources in an instance of* CSF.

Proof. We first show that CSF $\in$ NP. Given an instance of the problem and a set of trees $\mathcal{F}$ as a certificate (a solution to the problem), we can verify that the certificate is a capacitated spanning forest using a modified breadth-first search algorithm. For every tree in $\mathcal{F}$, begin the search from the tree's source while verifying at each step that all the conditions of the CSF hold: 1) Each tree is a tree (i.e. has no cycles), 2) Each tree contains exactly one source (i.e. is not connected to another tree), and 3) The cardinality of each tree does not exceed the capacity of the source. Finally we verify that each vertex in V is in a tree, which we can do by marking all nodes visited during the search, and verifying at the end of the search

that $V_{visited} = V$. This verification algorithm runs in polynomial time, thus the problem is in NP.

We now show that SAT $\leq_P$ CSF, that is, that SAT can be reduced to CSF in polynomial time. We are given as input an arbitrary boolean formula ϕ with m clauses and n variables. We convert this into an instance of the CSF problem so that this instance has a feasible solution if and only if ϕ is satisfiable. We construct a graph $G = (V, E)$ with $2n + m$ vertices and $n + 1$ sources. Our construction is similar to the one used in (Pap78b). For each variable x_i of ϕ we create two vertices x_i and $\bar{x}_i$ in G. We place an edge from x_i and $\bar{x}_i$ to each of x_{i+1} and $\bar{x}_{i+1}$ for $i = 1, \ldots, n$. For each clause $C_j \in \phi$ we create a vertex C_j and place an edge from C_j to each of its literals. We create n source vertices with capacity 1 and link each such source vertex s_i with edges to x_i and $\bar{x}_i$ for $i = 1, \ldots, n$. Finally, we create a source vertex s_0 with capacity $n + m$, and connect it to x_1 and $\bar{x}_1$. This completes the construction of G. An example of this construction for the boolean formula $\phi = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_3)$ is shown in Figure 10.

The intuition behind this construction is that the sources $s_1, \ldots, s_n$ function as "switches", each $s_i : i = 1, \ldots, n$ selecting either x_i or $\bar{x}_i$ by covering them in a tree of cardinality 2 (the literal and the source). This allows the tree rooted at s_0 to cover the complementary literals, which represents the truth assignment of ϕ. The 2^n possible assignments are super-polynomial, just as in SAT.

The construction of G can be done in polynomial time. We complete the proof by proving that this transformation of ϕ into G is indeed a reduction, that is, that there is a capacitated spanning forest in G if and only if ϕ is satisfiable, and vice versa.

Suppose $\mathcal{F}$ is a capacitated spanning forest on G. Then the tree T_0 rooted at s_0 will contain the vertices $C_1, \ldots, C_m$ corresponding to the clauses of ϕ, since it is the only tree with enough capacity to do so, all other trees being "switches" with a capacity of 1. Since the only way to reach a clause vertex is from a literal vertex linking to it, if T_0 contains a clause vertex, it must contain at least one of the literal vertices linking to it. Because $\sum_0^n c_i = |V|$, for $\mathcal{F}$ to be a spanning forest, all capacities must be used. Thus, since $c_0 = n + m$, T_0 must contain exactly m clauses and n literals. Since $c_1, \ldots, c_n = 1$, the remaining n literals are covered by switches. Thus, since any clause vertex is reachable only by a literal in T_0, if $\mathcal{F}$ exists, the n literal vertices in T_0 represent a satisfying assignment for ϕ.

Conversely, if ϕ has a satisfying truth assignment, this assignment

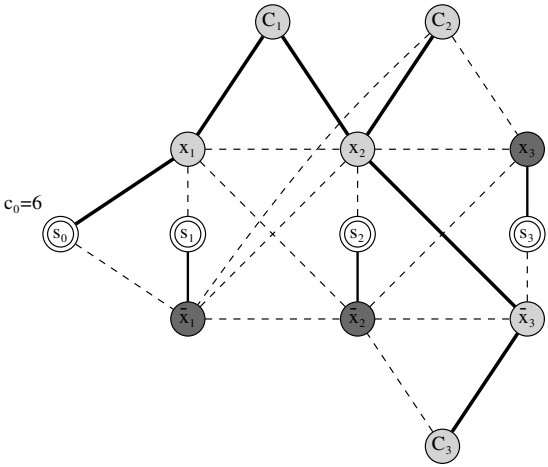


Figure 10: Proof of NP-completeness of CSF with n sources. CSF instance for the boolean formula $\phi = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_3)$. All non-source nodes are weighted 1. The satisfying assignment $x_1,\ x_2,\ \bar{x}_3 = 1$ corresponds to the tree rooted at s_0 shown in bold. Note that ϕ belongs to a hard case of SAT because it has one clause with three variables.

corresponds to the tree T_0 in G. For each true literal in the assignment, we add the literal's corresponding vertex in G to the new subgraph T_0, as well as the clause vertices and s_0. Since the nodes in this subgraph are connected by the construction of G, there must also be a spanning tree on the subgraph. Thus the edges in T_0 consist of the edges in G that form a spanning tree on $V(T_0)$. We would then need only to set $s_1, \ldots, s_n$ to cover the complementary literal vertices in order to create a spanning forest on G. □

4.2.2 Proof 2. CAPACITATED SPANNING FOREST is NP-complete for graphs with 2 sources.

Theorem 3. SAT $\leq_P$ CAPACITATED SPANNING FOREST *for graphs with* $|S| = 2$, *where* S *is the set of sources in an instance of* CSF.

Proof. Given an instance ϕ of SAT with n vertices and m clauses, we construct a graph similar to the one in the previous proof, and as illustrated in Figure 11. The graph has $2n+mn+2$ vertices, of which two are sources,

with capacities $c_0 = n + mn$ and $c_1 = n$ respectively. Each clause vertex C_i has edges to $n - 1$ vertices that we call "auxiliary vertices". The construction of this graph can be achieved in polynomial time.

We now prove that this is a reduction. We claim that there is a capacitated spanning forest on G if and only if ϕ has a satisfying truth assignment.

First, suppose that there is a capacitated spanning forest $\mathcal{F}$ on G. Then the tree T_0 rooted at s_0 will have $n + mn$ vertices and must include the vertices $C_1, \ldots, C_m$ plus their auxiliary vertices, since it is the only tree with sufficient capacity to do so. To demonstrate this, we first note that the only way to reach clause vertex C_j is from a literal vertex x_i, and likewise, the only way to reach the auxiliary vertices of C_j is through C_j. However, once C_j is part of a given tree, all the auxiliary vertices adjacent to C_j can only be covered by that tree since they are only reachable from C_j, which is now in the tree. Since the tree rooted at s_1 has a capacity of n, and it would require a capacity of at least $n + 1$ to cover C_j, its $n - 1$ auxiliary vertices, and the literal vertex x_i that led to C_j, the tree rooted at s_1 cannot cover any of the clause vertices if $\mathcal{F}$ is a spanning forest. Thus the only way for clause C_j and the auxiliary vertices to be covered is for them to be in T_0 having capacity $n + mn$, which is enough to cover n literals, and all of the m clauses and $m(n - 1)$ auxiliary vertices. As in the proof of Theorem 2, we consider that if a literal vertex x_i is covered by tree T_0 this corresponds to its assignment being 1 in ϕ. Since in a capacitated spanning forest all vertices are covered, and all clauses and n literals must be part of tree T_0, T_0 corresponds to a satisfying assignment in ϕ. The remaining n literals are covered by T_1 rooted at s_1.

Conversely, suppose that ϕ has a satisfying truth assignment A. Let the vertices corresponding to the literals in $\bar{A}$, the complement of that assignment, be assigned to tree T_1 rooted at s_1. There are n such literals and T_1 has a capacity of n, so the tree can cover them. Furthermore, by the construction of G, there is a path $s_1, l_1, l_2, \ldots, l_n$, where $l_i \in \{x_i, \bar{x_i}\}, i = 1, \ldots, n$ passing through every literal of $\bar{A}$, and connecting them to s_1.

The remaining vertices in $G' = G \backslash T_1$ are assigned to tree T_0. This consists of $n + mn = c_0$ vertices, corresponding to the selected literals in A, the satisfied clauses, and the auxiliary vertices. Since the sum of these vertices is equal to c_0, T_0 can cover these vertices. We know the subgraph G' is connected because removal of the path of literals in T_1 still leaves the other literals connected by the remaining edges to adjacent literals, i.e. if $(l_i, l_{i+1}) \in T_1, (\bar{l}_i, \bar{l}_{i+1}) \in G', i = 1, \ldots, n - 1$. Since none

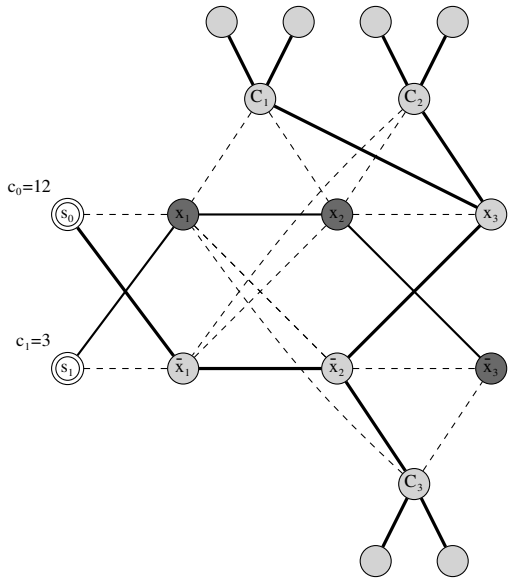


Figure 11: CSF instance for the 3CNF boolean formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$. All non-source nodes are weighted 1. The satisfying assignment $\bar{x}_1,\ \bar{x}_2,\ x_3 = 1$ corresponds to the tree rooted at s_0 shown in bold. The transformation from any 3SAT instance to a 2-source instance of CSF is: $\langle \{m \text{ clauses}\}, \{n \text{ variables}\}\rangle \implies \langle G = (\{2n + mn + 2 \text{ vertices}\}, E), \{2 \text{ sources}\}, \{c_0 = n + mn, c_1 = n\}\rangle$

of the clauses or the literals connected to the clauses were removed with T_1, the clause nodes and their auxiliary nodes are also in the connected graph G'. Taking any spanning tree on this subgraph (and a spanning tree must exist, since G' is connected) will produce a tree meeting the capacity requirement of T_0 (since $|V_{G'}| = c_0$). Thus given A we have a capacitated spanning forest on G. □

4.2.3 Proof 3. CAPACITATED SPANNING FOREST is NP-complete for sources with arbitrary capacity.

Theorem 4. SAT $\leq_P$ CAPACITATED SPANNING FOREST *for multiple sources of any capacity* > 1.

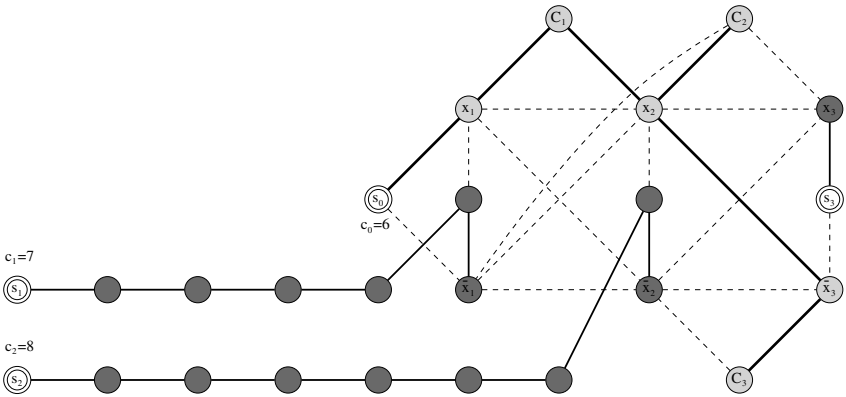


Figure 12: Proof of NP-completeness of CSF for sources with arbitrary positive capacities. All non-source nodes are weighted 1. The satisfying assignment of ϕ is covered by the tree of s_0. s_1 and s_2 have capacities > 1, however they are exhausted by the paths of dummy nodes until they reach their respective literals.

Proof. The argument follows the same reasoning as in Theorem 2. However this time each source s_i has a path of k_i : $k_i \in \mathbb{Z}^*$ vertices between it and its respective variable node pair $(x_i, \bar{x}_i)$, exhausting its capacity to 1 before x_i or $\bar{x}_i$ is covered. An example of this construction is shown in Figure 12. □

The same argument can be extended to the case in Theorem 2 to make the proof valid for graphs with $k \geq 2$ sources and arbitrary positive capacities.

4.3 Conclusion

In this section we have demonstrated the NP-completeness of the Capacitated Spanning Forest problem for cases where $k \geq 2$ and capacities are positive. This justifies the heuristic approaches taken in (DV17) and (DSV17).

Chapter 5

Network Planning in Smart Grids via a Local Search Heuristic for Spanning Forest Problems

Summary

In this chapter we present a Local Search heuristic algorithm for the CAPACITATED SPANNING FOREST problem. We demonstrate the performance of our algorithm on planar square grids and on graphs inspired by real-world urban distribution grid topologies for the problem of Network Planning, namely connecting the customers of a grid to energy sources. The work presented in this chapter has been published in (DSV17), and presents the first algorithm in the literature for CSF.

5.1 Introduction

As mentioned in Section 1.4, broadly speaking, Network Planning refers to the configuration of a power distribution network. In the context of

this work, we refer to Network Planning as the automated initial configuration of switches and lines in the electrical distribution grid to obtain a certain topology that connects the customers on the grid to a set of energy sources with given capacities. In essence, this is adapting the Capacitated Spanning Forest problem to a Smart Grid context, in setting up the initial topology of the grid. We refer to this phase of obtaining a solution to CSF on a Smart Grid's electrical network as the "Network Planning" phase. Due to their reconfigurability in contrast with traditional power distribution networks, smart grids allow for much greater flexibility in their planning (SXY$^+$16).

The work in this chapter addresses the problem of Network Planning in electrical distribution networks, namely the automated configuration of the network's topology to optimally assign customers to energy sources. In Chapter 2 we showed that in our model this reduces to the combinatorial optimization problem of constructing a spanning forest of rooted, vertex-disjoint, and weight-bound trees, which we termed the CAPACITATED SPANNING FOREST problem. We developed a Local Search heuristic that solves the CAPACITATED SPANNING FOREST problem and present our results in this chapter. The same heuristic can be applied to the problem of self-healing, which is addressed in the next chapter.

5.2 Model and problem description

As mentioned in Section 2.2, this work extends research by (QCS14), and we will use a similar model. Following the model in (QCS14), we represent an urban electrical distribution grid as an undirected graph in which nodes represent energy sources, customers, or other operational elements (switches, circuit breakers and feeders), while edges represent arcs (or lines) in the power network, including redundant dormant links that can be activated selectively. Each source is assigned a capacity indicating the amount of commodity (energy units) that it can provide to the network, while each customer has a demand in energy units. We refer to the graph with this information as the *underlying graph* of the network.

Figures 13 and 15 show representations of a grid in our model, including lines, energy sources with capacities, and customers with demands.

The motivation behind our use of undirected graphs is that the Smart Grids being modeled are supposed to be bi-directional grids supporting bi-directional flows to and from prosumers, meaning the network must also permit reversals of flows. Another reason is that the work that inspired this research by (QCS14) based its models of smart grids on undirected graphs. As such, we use the same model to continue their work. Several previous studies on the topological properties of electrical grids have also used undirected graphs as models (HBSB10) (HWR$^+$13) (SBP$^+$08) (CDH$^+$10) (KHGP13) (IKSW13). Another reason is that the directed and undirected versions of the CSF problems are different problems requiring different approaches; a parallel can be seen in the symmetric (undirected) and asymmetric (directed) Traveling Salesman Problems, for which there are different approximation algorithms and approximation ratios. For this work we have chosen to study the undirected version of the problem, which we consider more general. This is not to say that there is a need to impose a restriction on the graphs to be undirected in further studies. At present, we are not sure of the time complexity of CSF on directed graphs, but it is also likely NP-hard. We suspect that CSF is easier to solve using our approaches on undirected graphs rather than directed due to fewer restrictions.

In the Network Planning phase, our objective is to connect each customer to a source, such that the capacity of the source is not exceeded by the total demand of the customers. We do this by selecting edges in the underlying graph to construct a spanning forest of trees connecting each source to its customers. We refer to the tree formed by the source, its customers, and the set of edges connecting them as the *active tree*. Any unused edges from the underlying graph that are not in any active tree are dormant links that can be activated later as needed, either at the planning stage when there is a shift in demand or supply, or in the case of failure. In the Network Planning phase we seek to find a set of active trees that span all customer nodes of the graph.

While (QCS14) focused on cases with a single source, the problem of

cases featuring multiple energy sources was left open. When the graph has multiple sources, instead of constructing a single spanning tree, we must construct a *spanning forest* (see Fig. 8 on page 26) consisting of several active trees rooted at their respective sources, with each tree being disconnected from the others – that is, vertex-disjoint. The union of all trees should cover the entire set of customers. The trees are vertex-disjoint so that no customer may be served by more than one source. As in the single-source case, in the event of a disruption inactive links can be activated to recover connectivity.

As mentioned in Chapter 2, this problem of covering the vertices of a graph with a spanning forest consisting of k vertex-disjoint trees rooted at different nodes, with each tree having a capacity limiting the sum of weighted vertices that it can contain is the CAPACITATED SPANNING FOREST problem (CSF).

5.3 Contribution

In this work, we introduce a framework to represent a Smart Grid infrastructure and the CSF problem. We developed a Local Search heuristic for CSF that can be used to 1) construct a spanning forest, assigning customers to sources and 2) restore connectivity in case of perturbations in the graph (this result is shown in Chapter 6). Our Local Search heuristic is successful in constructing forests on planar grids and on a sample graph inspired by a real-world urban distribution network. We demonstrate the successful performance of this heuristic on several graph instances.

5.4 Local Search Heuristic

We developed a Local Search heuristic algorithm for solving the CAPACITATED SPANNING FOREST problem. Our approach is to first obtain an initial solution by expanding each tree concurrently according to a simple Breadth First Search (BFS) or Depth First Search (DFS) algorithm to yield an initial, typically suboptimal configuration of active trees. The

trees expand in a breadth-first or depth-first manner until no further nodes can be added, either because they had no remaining capacity or they are blocked by another tree as in Fig. 8b on page 26.

The simple initialization step in most cases is insufficient to guarantee a cover of the entire graph. This is usually due to one tree blocking another tree during their construction; the reader can refer to Fig. 15 in Section 5.5 for some examples of this. Thus, further processing is needed to improve the solution, for which we have developed a Local Search heuristic.

The main idea behind our algorithm is to, at each iteration, pass a random node from a tree T_1 that has exhausted its capacity (i.e. $|V(T_1) \setminus s_1| = c_1$) to one of its neighboring trees T_2 that is not yet at capacity, followed by assigning a new, unserved neighboring node to T_1. These steps are performed until either all nodes are served, all trees are at capacity, or the maximum number of iterations has been reached, which we define as $|V| \times 2$.

The reasoning behind this algorithm is that a tree that has reached its capacity limit and has neighbors that are not yet at capacity is probably blocking those neighboring trees, as in Fig. 8. Through these node exchanges the unsatisfied neighboring tree can slowly grow, however gaps in blockages that might emerge can also be exploited.

Once the node exchange is made, all trees in the graph expand concurrently according to the heuristic they used in the initialization step (Breadth-First or Depth-First Search). This exploits newly-created gaps in blockages so that tree expansions can occur more efficiently, rather than only one node at a time.

In the single node exchange step of the heuristic, the node being given up from the full tree is preferably a leaf node. This is because changes in non-leaf nodes disconnect from the source all nodes that are downstream of the change location. This is avoided by exchanging only leaves. If a leaf node cannot be exchanged because T_2 is not in contact with any of T_1's leaves, we exchange any random node v from T_1 adjacent to the recipient T_2, which will also disconnect several nodes in T_1 downstream of v. These nodes can be recovered in the expansion step.

The heuristic is stated more formally in Algorithm 1.

Algorithm 1 Local Search Heuristic

Require: Graph is initialized with simple BFS or DFS
1: **while** $\forall T \in \mathcal{T},\ T$.capacity $\neq 0$ **and** $|V_{\text{uncovered}}| \neq 0$
   and $iterations < |V| \times 2$ **do**
2: $T_1 \leftarrow$ random $T \in \mathcal{T}$ with $T.capacity = 0$
3: $T_2 \leftarrow$ neighbor_tree(T_1) having $T_2.capacity > 0$
4: $v \leftarrow$ random $v \in leaves(T_1)$ adjacent to T_2
5: **if** $leaves(T_1)$ adjacent to $T_2 = \emptyset$ **then**
6: $v \leftarrow$ random $v \in T_1$
7: **end if**
8: Give v from T_1 to T_2
9: $u \leftarrow u \in neighbors(T_1)$ where u is unserved
10: $T_1 \leftarrow T_1 \cup u$
11: **for all** $T \in \mathcal{T}$ **do**
12: T.expand()
13: **end for**
14: **end while**

The structure of the algorithm suggests that its time complexity is polynomial with respect to the input size. In the worst case, assume a case with two trees T_1 and T_2 with initial capacities c_1 and c_2 where $c_1 = c_2 = \frac{|V|}{2}$. Suppose after the initialization step that $T_1.capacity = 0$ and $T_2.capacity = \frac{|V|}{2}$. Suppose that this situation holds after every iteration of the algorithm and the trees are in constant contact at all times, in other words, that after each iteration of the algorithm one node is exchanged at a time from T_1 to T_2. Then in total $c_2 = \frac{|V|}{2}$ node exchanges will need to take place in the worst case; in other words, assuming two trees, $O(c_i)$ is the expected worst case running time, where c_i is the capacity of the least satisfied tree.

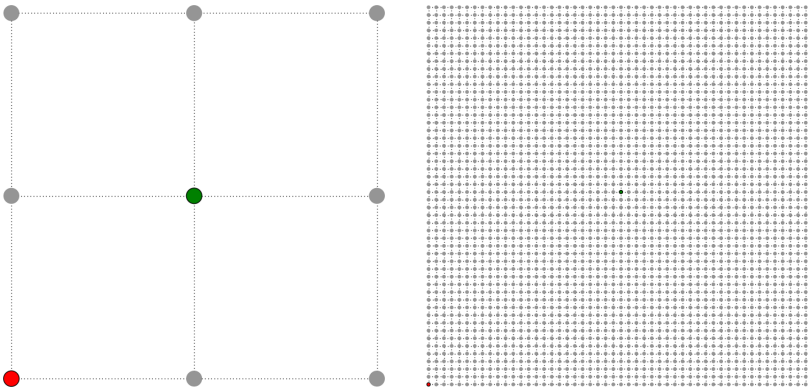
5.5 Experimental Simulations

5.5.1 Experimental setup

To verify the performance of our heuristic algorithm experimentally, we implemented a test bed to simulate distribution networks in Python (Ros95) using the NetworkX module (HSS08). The graphs we modeled in our test bed were planar square grid graphs of sizes $N = 9$, 2500, 10,000 and 20,000 nodes (Figs. 15 and 17) and we also modeled the topology of a "real-world" urban distribution network, that was introduced in the FLISR case study scenario shown in Fig. 5 in Section 1.4. In all graphs, the demands of customers were uniformly set to 1. In the real-world graph, customer nodes have a demand of 1, while other nodes (remote operated switches, circuit breakers, feeders and junctions) have a cost of 0. The roots of each tree (sources) were each assigned a given capacity, such that in summation they can supply the entire graph. Simulations were performed where the trees concurrently expanded from their root nodes via the successive addition of other nodes.

We performed experimental simulations to test the heuristic's performance, averaging the results of 1000 independent simulation runs over several test scenarios. The variables manipulated in the test scenarios were the type of underlying graph (square grid or sample electric grid), the type of algorithm used to obtain an initial partial forest (BFS or DFS), and in the square grid case, the size of the graph (9 nodes or 2500 nodes). The metric used to evaluate the algorithm's performance was the fraction of customers served, or *FoS*, defined as $FoS = \frac{|\text{served nodes}|}{|\text{total nodes}|}$.

In our general experimental setup, we began with an underlying graph consisting of unassigned nodes and two or more sources, each source being assigned a capacity (Fig. 13). We then constructed an initial solution by applying a Breadth First Search or Depth First Search algorithm to grow trees outwards from each source, obeying the constraints of the CSF problem (for example that trees cannot share the same vertex). Figs. 15a, 15c and 15e illustrate the construction phase and the initial partial forest thus obtained. Besides an initial solution, this partial forest also serves as an experimental control and a benchmark against



(a) Initial $N = 9$ graph **(b)** Initial $N = 2500$ graph

Figure 13: Initial states of planar grid graphs, $N = 9$ (13a) and $N = 2500$ (13b). In both cases Source 1 (Red) is placed in the bottom left corner, and Source 2 (Green) is placed in the center. Each source has capacity $\frac{(N-2)}{2}$, while each customer has demand 1.

which the performance of the Local Search heuristic can be compared, since simple BFS and DFS algorithms typically do not yield an optimal cover. Once the initial partial forest was obtained, we executed the Local Search heuristic and recorded the resulting *FoS*.

For the square grid cases we used two sources with equal capacities, such that $\sum c_i = |V| - 2$. One source was placed in the center of the grid and another in a corner. In this configuration the simple algorithms used to initialize the graph were insufficient to produce an optimal covering, due to one source blocking off another (Figs. 15a and 15c). When both sources are placed in corners, it was found that BFS is sufficient to cover the graph.

We performed 1000 runs of simulations on square grids of size $N = 9$ and $N = 2500$ initialized with BFS and DFS. In addition, we also ran tests on graphs of size $N = 10000$ (Fig. 16) and $N = 25000$, and on graphs with three sources (Fig. 17).

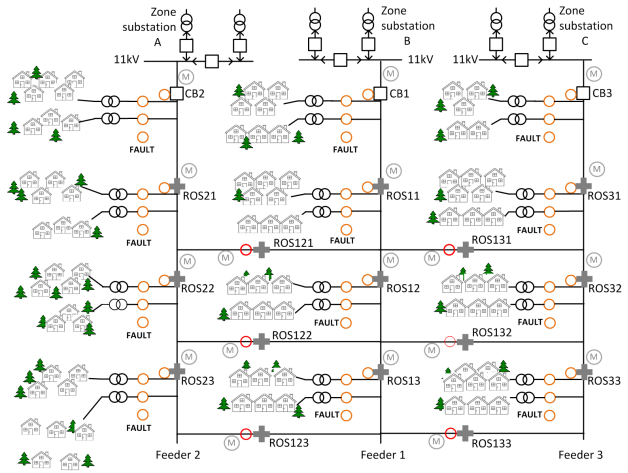
To simulate a "real-world" electric grid, we created within our framework a representation of the FLISR case study introduced in Fig. 5 from

Section 1.4, which we refer to as the *sample grid*. We performed a translation of all elements: sources, customers, switches, circuit breakers, and connective elements (Fig. 14). In the sample grid, we modeled three sources and 24 customers. Sources were all assigned the same capacity, which together was sufficient to cover the entire set of customers. All customers were assigned a demand of 1, while nodes representing switches, circuit breakers, junctions and feeders were assigned 0 demand, meaning they can be added to a tree at zero cost. Junctions or intersections in the electric lines were represented by a node connecting the relevant links. We did not model the physical limits of the grid.

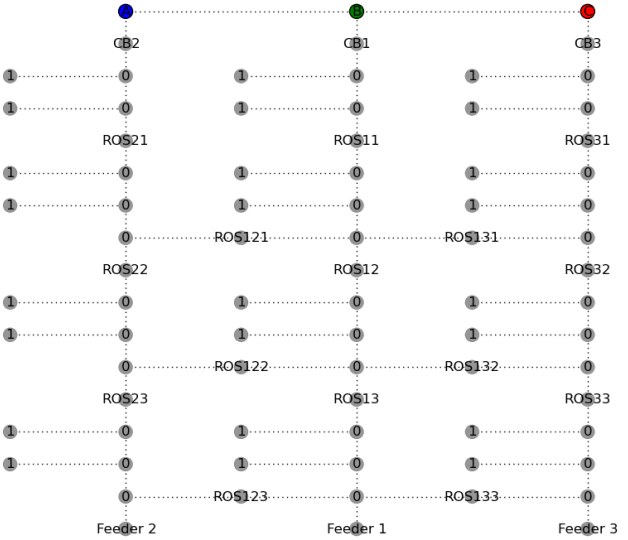
As in the square grid case, we initialized the graph using Breadth First Search or Depth First Search. Since not all nodes in this graph are customers, the BFS and DFS heuristic was modified to give preference to adjacent customer nodes before adding other types of nodes to the tree. We performed 1000 runs of simulations initialized with BFS and DFS and recorded the *FoS*. Only customer nodes are counted towards the *FoS*.

5.5.2 Experimental results

The results of our simulations show that the heuristic successfully constructs a capacitated spanning forest on both square grid and real grid topologies; in our trials a complete covering of the graph was obtained in the vast majority of cases in all test scenarios. Table 1 presents the results of the Local Search algorithm, averaged over 1000 runs of experiments for each experimental scenario.



Original



Translation

Figure 14: *Top:* Sample grid topology. *Bottom:* The translation in our platform. Junctions, relays and remote operated switches are represented as nodes with zero cost.

			FoS	
Graph	BFS	DFS	LS_{BFS}	LS_{DFS}
9 node grid	0.78	0.91	1.0	1.0
2.5K node grid	0.91	0.56	1.0	0.99
Sample Grid	0.55	0.76	0.99	1.0

Table 1: Summary of mean *FoS* results over 1000 runs. Columns BFS and DFS show results with those algorithms alone, while columns LS_{BFS} and LS_{DFS} show results from the Local Search heuristic on graphs initialized with those respective algorithms. The difference in performance after adding the Local Search heuristic are very clear and no statistical test is required to show that the heuristic improves significantly over the initial solutions.

Figures 15–18 show sample runs demonstrating the performance of the Local Search algorithm in constructing a spanning forest on square grids and the sample electric grid. The heuristic successfully constructs a forest even when it is initialized with very unfavourable starting conditions, such as when initializing with DFS, which often resulted in one source being completely surrounded by another and zero nodes being added to the surrounded source (Fig 15f). The Local Search heuristic is able to recover and reach the optimum also in this case.

More specifically, Figures 15a and 15b show a graph with two sources (in green and red) on a 3x3 grid, the first in the center and the second in the bottom left corner. The sources have enough capacity to cover all the nodes together, namely capacities 3 and 4 respectively. Figure 15a shows the configuration of the trees following initialization with Breadth-First Search. The tree formed by the central green source blocks the red source in the bottom left corner. Figure 15b shows the state after the Local Search is executed, with all nodes being covered.

Figures 15c and 15d show the same type of experiment, but on a larger 50x50 grid with 2500 nodes. Since there is more space between the sources, they have more room to expand by BFS before becoming stuck. Here again, after a few node exchanges, the Local Search heuristic creates a gap through which the bottom left source can expand to the unserved nodes.

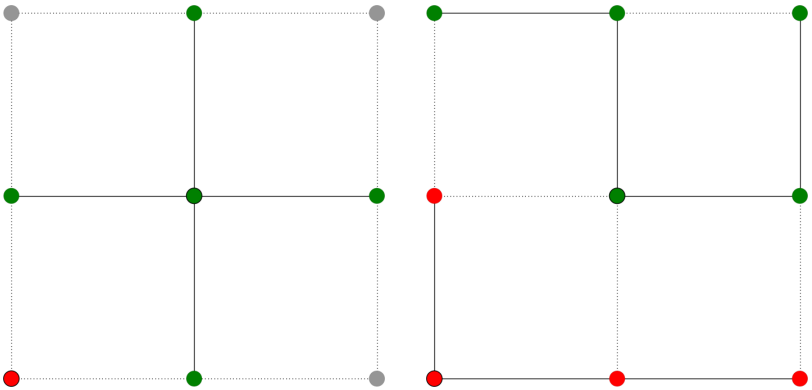
Figure 15e 15f show the same 50x50 grid setup, but using Depth-

First Search in the initialization phase. Here the bottom right source is blocked at initialization without adding any nodes. Nevertheless the Local Search heuristic, through several node exchanges, manages to cover the entire graph. Together these results demonstrate that the Local Search is successful in modifying several types of initial configurations of the graph.

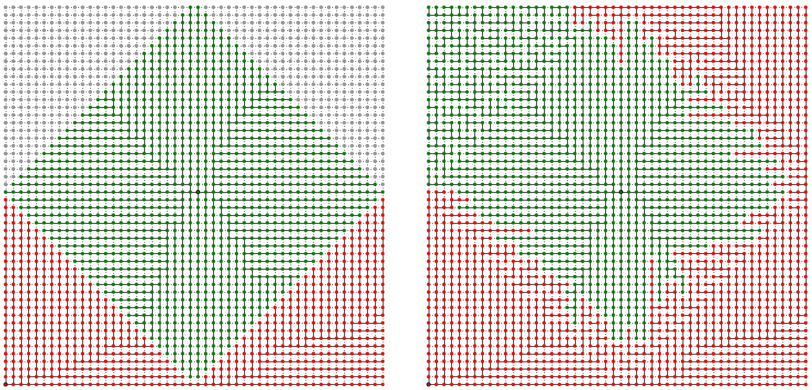
Although we did not perform a large number of experiments on larger graphs, we verified experimentally using fewer runs that the heuristic is equally effective on grids of size $N = 10,000$ (Fig. 17) and $N = 20,000$.

An explanation for the effectiveness of the heuristic is that the algorithm modifies the graph at each iteration by at least one node (Operations 9 and 10 in Algorithm 1), adding it to a tree at capacity and giving up a node to a neighboring unsatisfied tree. Through these minuscule but persistent changes, whose "footprint" can be seen in the structure of the final graph, an optimal solution is more likely to be reached than through larger perturbations such as the disconnection of several nodes at once, a step which may repeatedly miss the optimum. The cost of making these many single-node changes is in the larger time taken by the algorithm to converge.

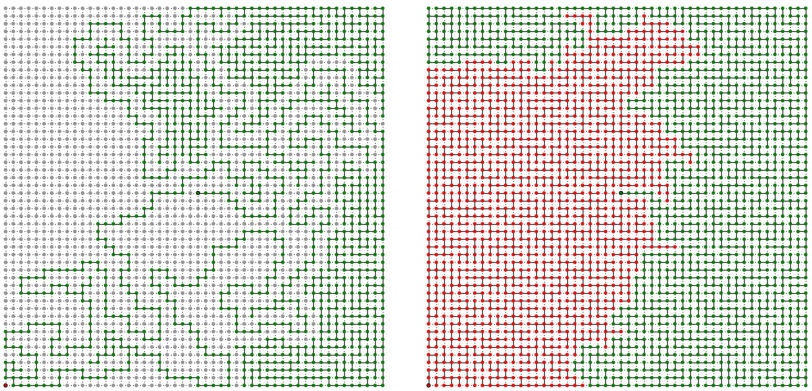
Although successful in the vast majority of cases, our heuristic does not guarantee success 100% of the time. A possible explanation for this is that the maximum number of iterations was reached in some tests; we believe that given a bit more time the heuristic could eventually reach a solution.



(a) BFS **(b)** Local Search Heuristic



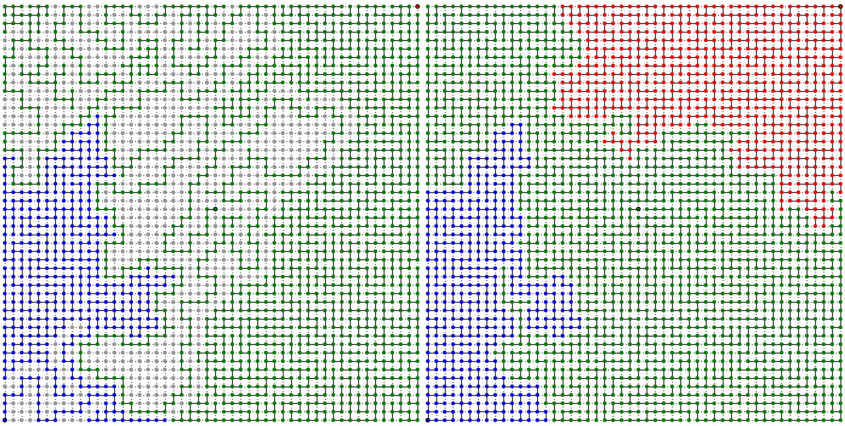
(c) BFS **(d)** Local Search Heuristic



(e) DFS **(f)** Local Search Heuristic

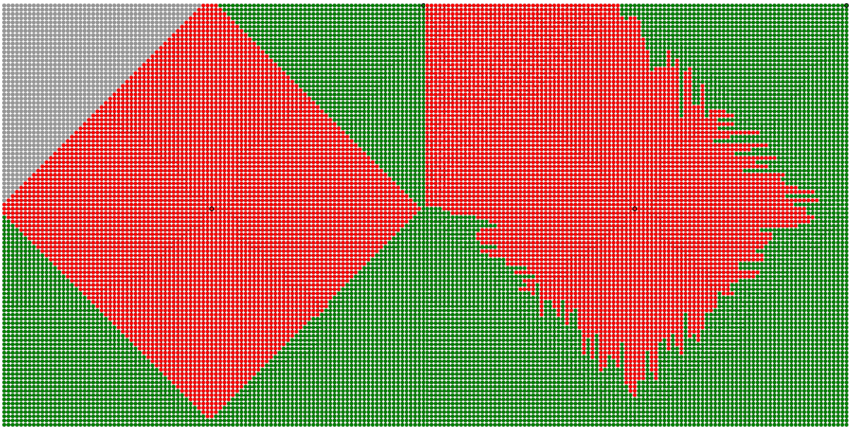
Figure 15: Sample results of Local Search on grids of size $N = 9$ and $N = 2500$. Initialization step is shown in left column, Local Search in right column. When the graph is initialized with BFS (Figs. 15a and 15c) or DFS

(Fig. 15e), Source 1 (Red) is surrounded or blocked by Source 2 (Green), resulting in many starving nodes (grey). Applying the Local Search algorithm leads to an optimal covering of the graph on all three starting graphs. Of note is that the Local Search manages to construct a forest even when Source 1 initially has no nodes at all, and is completely surrounded by Source 2, as in Fig. 15e.



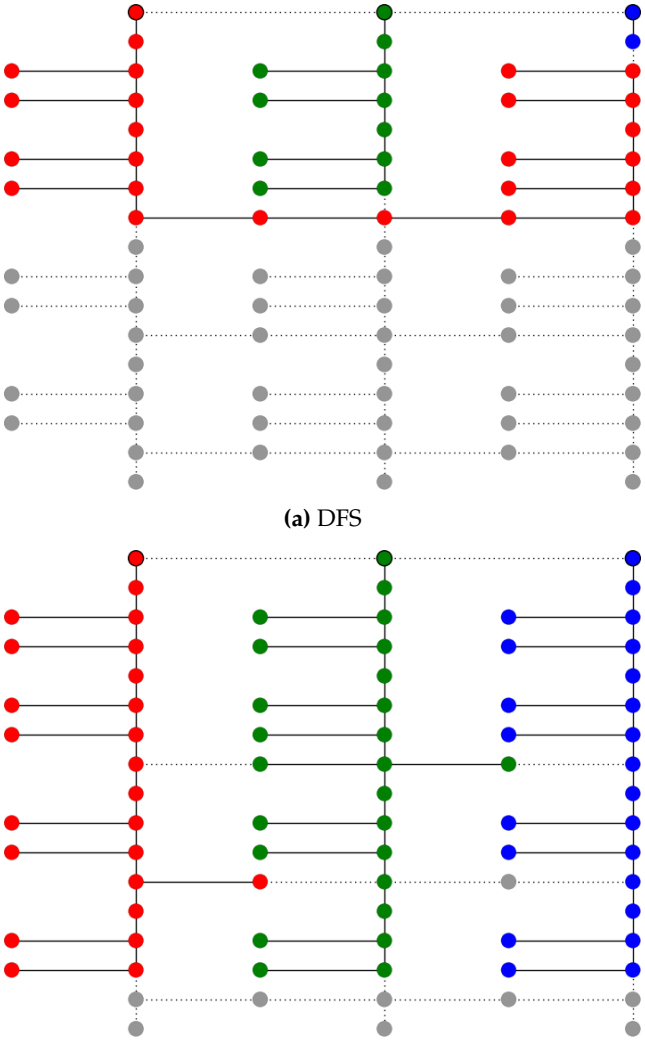
DFS Local Search Heuristic

Figure 16: Local Search on N = 2500 grid with 3 sources (blue, green, red)



BFS Local Search

Figure 17: Local Search on N = 10,000 node grid



(a) DFS

(b) Local Search Heuristic

Figure 18: Sample result of planning stage (i.e. assignment of customers to sources) on a real-world-inspired FLISR graph. Graph is initialized with DFS in 18a. Heuristic allocates all customers to a source in 18b.

Chapter 6

Network Planning and Self-Repair in Models of Urban Distribution Networks via Hill Climbing

Summary

In this chapter we present a heuristic algorithm for the CAPACITATED SPANNING FOREST problem based on a Hill Climbing metaheuristic. This algorithm outperforms the Local Search heuristic from Chapter 5 in solution quality on more complex graphs, and can solve Network Planning as well as self-healing scenarios on large graphs representing smart grids. The work presented in this chapter is published in (DV17).

6.1 Introduction

Network Planning, as well as Fault Location, Isolation and Supply Restoration (FLISR) are both important functions of power distribution automation systems. As mentioned in Section 1.4 and Chapter 5, Network Planning broadly refers to the configuration of a power distribution network,

while Fault Location, Isolation and Supply Restoration (FLISR) deals with repairing faults on power distribution lines. In this chapter we introduce an algorithm that addresses both problems.

The work presented in this chapter extends the results presented in Chapter 5 and (DSV17), wherein a Local Search heuristic was used to solve a network planning problem of assigning customers to energy sources in smart grids. We introduce an algorithm based on a Hill Climbing (HC) metaheuristic that embeds the Local Search from (DSV17), and that outperforms the plain Local Search on larger graphs. We demonstrate experimentally the performance of the heuristic on large graphs inspired by the topologies of real electric grids through thousands of runs of simulations. The graphs we test on are expansions of the "sample grid" introduced in (DSV17) that is inspired by real-world networks from the Australian supplier ENERGEX (HVNS11). This graph is based on the topology of the FLISR scenario described in Section 1.4, illustrated in Figure 5. We demonstrate also the use of the Hill Climbing heuristic for the problem of fault restoration when parts of the grid are disconnected.

The main motivations for this work were twofold. First, the Local Search introduced in (DSV17) was found to not perform adequately on the aforementioned expansions of the "sample grid" graph for the network planning problem. Second, following up on the results and future work mentioned in (DSV17), we aimed to tackle the problem of repairing a network in the case of link or node failures (the Fault Location, Isolation and Supply Restoration, or FLISR problem).

6.2 Problem description and model

In the context of this work, Network Planning refers to the automated initial configuration of the network's topology in order to optimally assign customers to a set of energy sources, each with a given capacity. An envisioned feature of smart grids is self-repair in the case of failures. Fault Location, Isolation and Supply Restoration (FLISR) broadly refers to the repair of faults on power distribution lines. An example of a FLISR scenario is illustrated in Fig. 5 reproduced in Section 1.4 from (ZVD15).

We define self-healing as the automated restoration of connectivity following a link or node failure that disconnects customers from the grid. This is achieved by exploiting the redundant "dormant" links already present in the distribution network after the network planning phase, activating them as needed to reconnect disconnected nodes. Figure 6 from Section 2.2 illustrates this concept. This model is rooted in reality as urban low-voltage distribution networks already contain redundant links that can be activated to restore connectivity when failures occur. These redundant links exist in the form of tie switches between different feeders, which can be activated or deactivated by opening or closing the switch.

Following the model introduced in (QCS14) and used in Chapter 5, we represent the electrical distribution grid as an undirected graph in which nodes represent energy sources, customers, or other operational elements (switches, circuit breakers and feeders), while edges represent arcs in the power network. Each source is assigned a capacity indicating the amount of commodity (electric power) that it can provide to the network, while each customer node has a demand in energy units. In the current model, there are no capacity constraints on the lines (edges), only on the sources.

In the Network Planning phase every customer is connected to a source, such that the sum of its customer demands does not exceed its capacity. In the Self-Healing phase, we are given a graph of a spanning forest that has suffered some perturbation such as the removal of a node or edges and must reconnect disconnected customer nodes to sources. Failures in the network can occur either as link failures that remove an edge from the graph, or node failures that remove a node and all its adjacent edges.

In (DSV17) and Chapter 2 we described the CAPACITATED SPANNING FOREST problem (CSF) as the problem of creating on a graph a spanning forest of k vertex-disjoint trees rooted at distinct nodes, with each tree having a capacity that limits the sum of weighted vertices that it can contain. Both Network Planning and Self-Healing can be modeled as instances of the CAPACITATED SPANNING FOREST problem. The Network

Planning phase deals with constructing a capacitated spanning forest on a graph representing a power grid, while in the Self-Healing phase, we are given a graph of a spanning forest that has suffered some perturbation and must reconnect disconnected customer nodes to sources, reconstructing the spanning forest.

6.3 Algorithm

6.3.1 Initialization

In (DSV17) we introduced in a Local Search heuristic algorithm for CSF. An initial solution is obtained by building the trees in the graph according to Breadth-First or Depth-First Search until moves are no longer possible. As this initial forest is usually suboptimal it is improved upon by a Local Search heuristic. In (DSV17), the algorithm used to initialize the graph was a Breadth-First Search or Depth-First Search, executed concurrently for each tree in separate threads. Because fairness was not guaranteed, in the current work we used a new algorithm that ensures trees make an equal number of moves at initialization. Each tree adds a node from its periphery at random, taking turns. This is implemented in the form of a list from which trees that have made a move are removed. Once all trees have made a move, the list is repopulated. This method was an improvement and yielded better initial solutions, especially in graphs with bottlenecks and choke-points such as the sample grids we studied, because trees are less likely to cut each other off early. Indeed, in some cases, such as the 4-source graph, it was even frequently possible to obtain the optimal configuration of trees just with this simple initialization procedure.

Algorithm 2 Initialization algorithm

$List \leftarrow \{T_1 \dots T_n\}$
repeat
  $T_i = List.pop()$
  $v \leftarrow$ random unserved node neighboring T_i
  $T_i \leftarrow T_i \cup \{v\}$
  if $List = \emptyset$ **then**
    $List \leftarrow \{T_1 \dots T_n\}$
  end if
until No more moves possible

This typically leads to a suboptimal solution. Once the graph has been initialized, we proceed with the hill climbing heuristic procedure.

Note that in our simulations, to test the performance of the Hill Climbing heuristic only, we omitted runs in which the initialization algorithm was sufficient to reach an optimal solution (which is the case for smaller graphs).

6.3.2 Hill climbing heuristic

Once an initial solution is obtained, the Hill Climbing heuristic in Algorithm 3 is applied to improve the solution. This algorithm consists of a main loop in which at each iteration a small local change (i.e. an exchange of nodes between two adjacent trees) is made on the current state of the graph. If the resulting solution is better or at least as good as the current solution, it is accepted and becomes the current solution; otherwise it is discarded and a new local change is tried.

Algorithm 4 describes the Local Search step performed. This simply consists of a random donation of one node from tree T_i to its neighbor T_j. The choice of T_i is preferential, first from any tree that is at full capacity, then if no such tree is available from any tree with unserved neighbors, and failing that, from any random tree. T_j is chosen from a random tree adjacent to T_i. Once node v is donated from T_i to T_j, T_i obtains a random neighboring unserved node, and all trees are then "expanded" according to the procedure in Algorithm 5. Some optimization has been made to

the overall procedure by omitting already discarded moves. The criterion being compared between solutions to determine acceptance is the Fraction of Serviced customer nodes (*FoS*).

Algorithm 3 Hill Climbing heuristic

Require: Initialize graph with Algorithm 5
$s_0 \leftarrow InitialSolution$
$s \leftarrow s_0$
repeat
$s^* \leftarrow LocalSearchStep(s)$
if $s^*.FoS \geq s.FoS$ **then**
$s \leftarrow s^*$
end if
until All customers served $\vee$ maximum iterations reached
return s

Algorithm 4 LocalSearchStep(s)

$$T_i \leftarrow \begin{cases} \text{random full tree} \\ \text{random tree with unserved neighbor} \\ \text{random tree with } |T| > 1 \end{cases}$$
$T_j \leftarrow$ random tree neighboring T_i
$v \leftarrow$ random node of T_i
$T_i \leftarrow T_i \setminus \{v\}$
$T_j \leftarrow T_j \cup \{v\}$
$u \leftarrow$ random unserved node neighboring T_i
$T_i \leftarrow T_i \cup \{u\}$
$expand(T_j)$

6.4 Experimental Scenarios

6.4.1 Experimental setup

To test experimentally the performance of the algorithm, we expanded the test bed used in (DSV17) that was implemented in Python (Ros95) using the NetworkX (HSS08) module.

We modeled graphs that were expansions of the "sample grid" graph in (DSV17). The sample grid graph is representative of real-world electric distribution networks and their topologies; it is illustrated in Figure 5 from Section1.4. This graph is based on the topological profile of urban 11 kV feeders from the Australian utility ENERGEX (HVNS11), but shown in simplified form, with only the backbones and ties to adjacent feeders represented; second and third-level spurs that are part of the network are omitted. The original graph has only three sources and 24 customer nodes, arranged on 3 feeders. The new graphs based on this model, illustrated in Figs. 19 and 20 are larger, featuring more feeders and junctions at remote operated switches. Essentially, the graph has been expanded to feature longer feeders and more feeders, with more junctions between them. Thus these graphs represent more complicated instances of the Capacitated Spanning Forest problem on a sparse graph. The expanded graphs have four and ten sources respectively and 128 and 800 customer nodes. The demands of customers were uniformly set to 1 as in the previous work, while other nodes such as circuit breakers, junctions and tie switches had no demand but could be added to the tree of a source.

We performed individual simulation runs and averaged the results of 1000 independent simulations for each test scenario. The metric used to evaluate the algorithm's performance was the fraction of customers served, or *FoS*, defined as $FoS = \frac{|\text{served nodes}|}{|\text{total nodes}|}$.

In our setup, we begin with an underlying graph consisting of unassigned nodes and multiple sources. Each source is assigned a capacity (Fig. 19). An initial solution is then constructed with Algorithm 5, growing trees outwards from each source, while obeying the constraints of the CSF problem (for example that trees cannot share the same vertex). Once the initial partial forest was obtained, we executed the Hill Climbing heuristic and recorded the resulting *FoS*.

6.4.2 Application 1: Network planning with a realistic layout

The first application explored is network planning on an enlargement of the sample graph scenario introduced in (DSV17), as shown in Figs. 19 & 20. In this application, the number of feeders (and corresponding sources) and the length of the feeders is extended. Network planning refers to the assignment of customers to sources such that each customer node is served by one source. This is achieved by building a capacitated spanning forest on the underlying graph. Larger graphs are more difficult instances for the heuristic to solve.

6.4.3 Application 2: Fault restoration

The second application we investigated is service restoration during the occurrence of a fault. When a fault occurs (such as through failure of a single link or node with several links connected to it), customer nodes might be disconnected from their energy source. Service restoration involves activating redundant dormant nodes to reconnect the customers to a source.

In our experimental setup for this scenario, the graph is initially fully covered by executing the Network Planning phase. The sources are given excess capacity (their capacity is doubled) to be able to serve additional nodes that have been disconnected. In our experiments, a single link is severed in the graph that is either randomly selected or specified. This has the consequence of disconnecting nodes that are downstream from it. The Hill Climbing heuristic is then executed in order to restore connectivity.

6.5 Results

6.5.1 Planning

Figure 19 shows sample results from our method for the network planning scenario on a grid with 4 sources and 412 nodes. After initial-

ization with Algorithm 5, some nodes (in grey) remain unassigned, so $FoS = 0.72$. Upon execution of the Hill Climbing heuristic, all customers are assigned to a source ($FoS = 1$). This experiment is repeated with 10 sources and a larger graph of 2590 nodes in Figure 20, and again the heuristic is successful.

Table 2 encapsulates the results of 1000 simulations on these two types of graph with the Local Search heuristic introduced previously in (DSV17) and the new Hill Climbing heuristic. Although adequate for the types of graphs presented in (DSV17), the Local Search heuristic fails to converge to a solution on these larger sample grids – in fact, it makes the initial solutions worse, resulting in a negative average ΔFoS for both 4-source and 10-source grids.

In contrast, the Hill Climbing heuristic is successful for grids of both sizes, yielding an average FoS of nearly 1.0. However, there is a marked increase in the number of iterations needed to reach a solution with respect to the size of the graph.

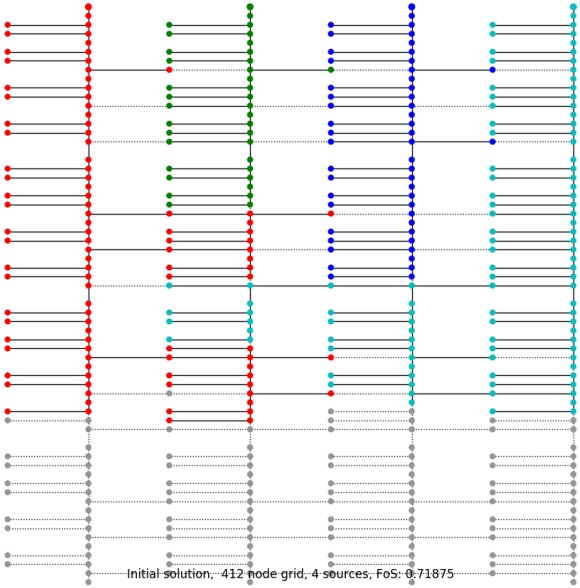
Table 2: Summary of mean FoS results over 1000 runs for graphs with 3, 4 and 10 sources. LS denotes Local Search heuristic from (DSV17) while HC is the new Hill Climbing heuristic.

Method	N	No. of sources	$\text{FoS}_{\text{final}}$	ΔFoS	iterations
LS	72	3	1.0	0.083	10.86
HC	72	3	1.0	0.083	7.55
LS	412	4	0.57	-0.26	4928
HC	412	4	1.0	0.17	514
LS	2590	10	0.31	-0.29	14702
HC	2590	10	0.996	0.392	63666

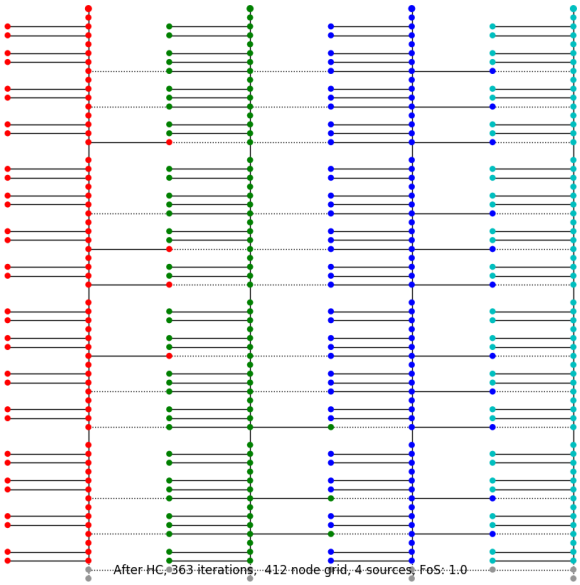
6.5.2 Fault restoration

Figures 21, 22, and 23 show sample runs of the application of the Hill Climbing heuristic in a repair scenario.

Figure 21 shows the action of the algorithm in the case of the single link failure and FLISR scenario described in Figure 5, on a graph with 24 nodes. A specified edge is removed from the graph and several nodes

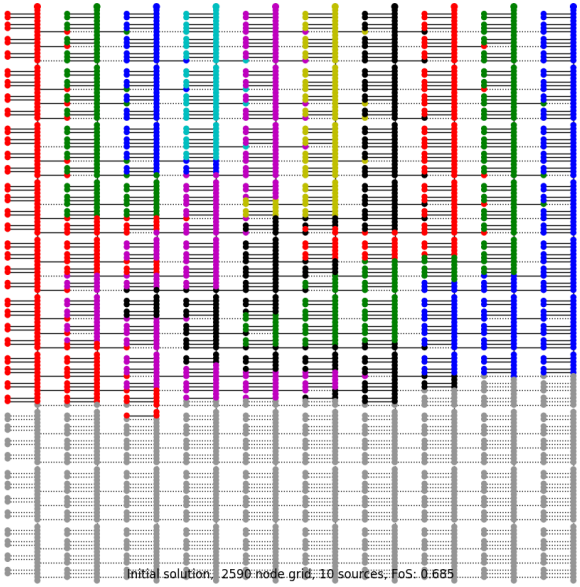


(a) Initial 4 source graph

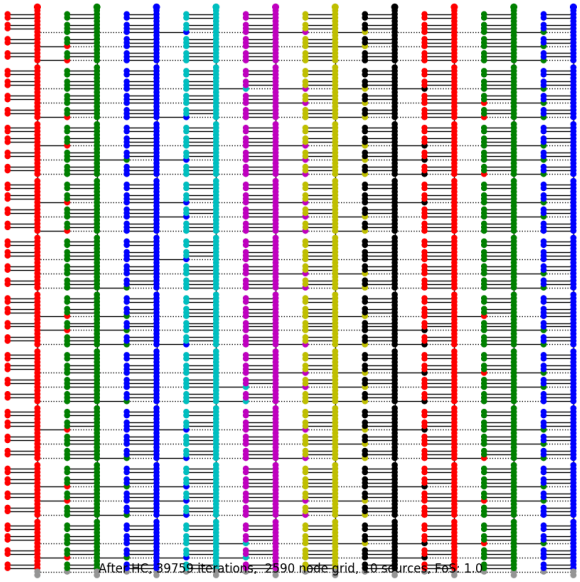


(b) Final 4 source graph

Figure 19: Network planning on an 412-node graph with 4 sources.



(a) Initial 10 source graph



(b) Final 10 source graph

Figure 20: Network planning on an 2590-node graph with 10 sources.

become disconnected from their source. Other sources with excess capacity are connected to the unserviced nodes to supply them, and the *FoS* is restored to its optimal value. This scenario can also be solved by the Local Search heuristic from Chapter 5.

Figure 22 shows a sample run of the heuristic for the repair scenarios on an expansion of the FLISR scenario sample grid graph from ENERGEX. This graph has more feeders customers and ties than the previous, essentially having the feeder elements repeated. The heuristic successfully restores connectivity to all customers (FoS $= 1$).

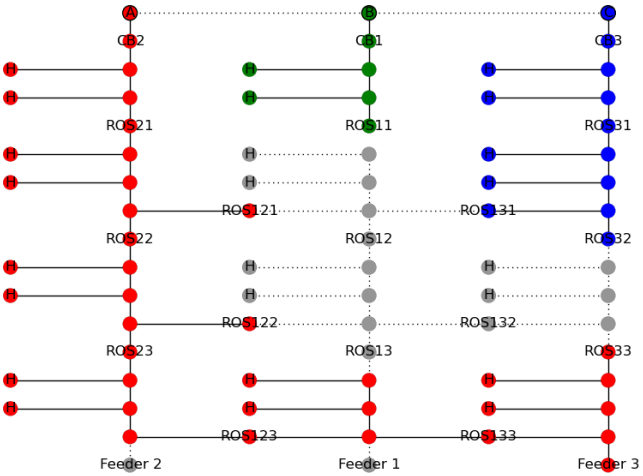
Figure 23 shows a sample run of the heuristic for the repair scenarios on the larger, more complicated 10-source graphs. The heuristic successfully restores connectivity to all customers (FoS $= 1$).

Although we performed several sample runs to test the Hill Climbing heuristic in repair scenarios that indicate to us that it can solve such problems, we did not perform a large number of experiments testing the *FoS* results of the Local Search vs the Hill Climbing heuristic for the repair scenarios. Further research is needed in this area. Our sample runs indicate that the Hill Climbing heuristic produces better quality solutions on average and with less execution time in the real-world repair scenarios.

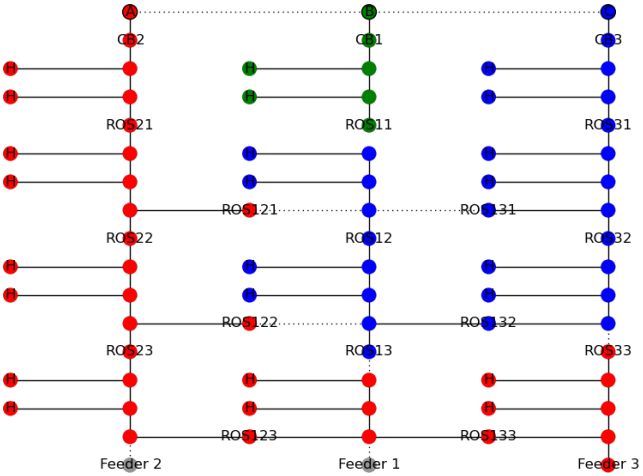
Figure 23 shows a sample run of the heuristic for the repair scenarios on the larger, more complicated 10-source graphs. The heuristic successfully restores connectivity to all customers (FoS $= 1$).

6.6 Conclusion

In this work we presented a Hill Climbing heuristic that can be used for network planning and grid self-repair. Our empirical findings suggest that this heuristic is effective on larger and much more complicated graphs than the heuristic presented in (DSV17) for constructing constructing spanning forests of capacitated, vertex-disjoint trees.

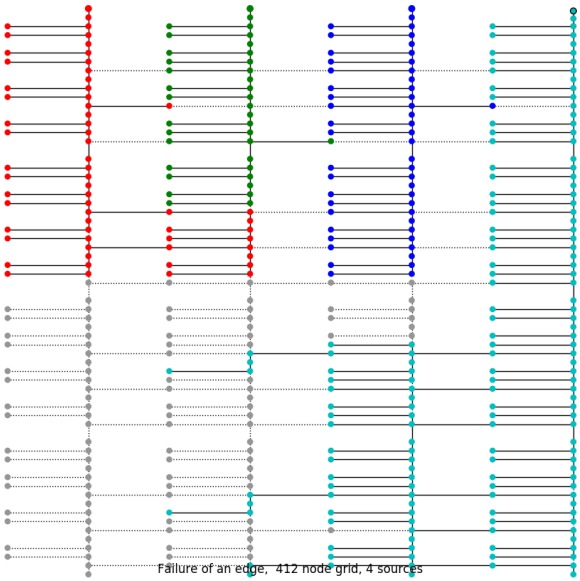


(a) Link failure

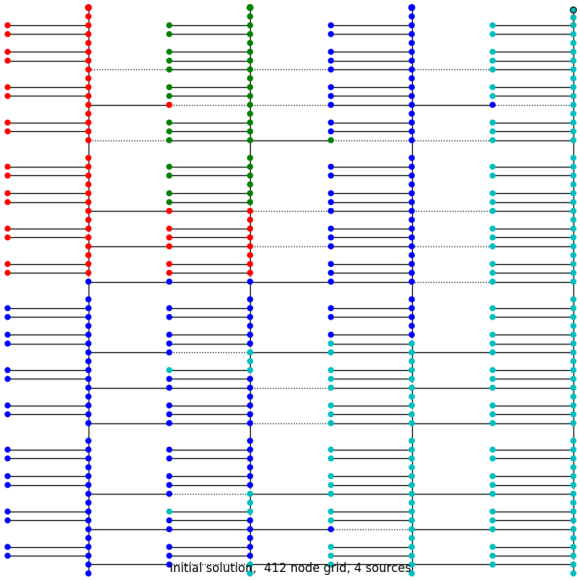


(b) Repair

Figure 21: Result of the repair algorithm. An edge failure occurs in tree B, causing loss of service in some nodes (in grey). Unserviced nodes are reassigned to sources with excess capacity.

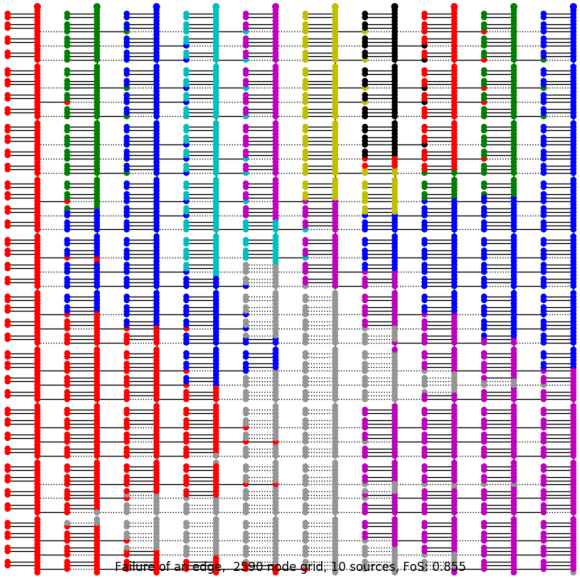


(a) Link failure - disconnected nodes shown in grey

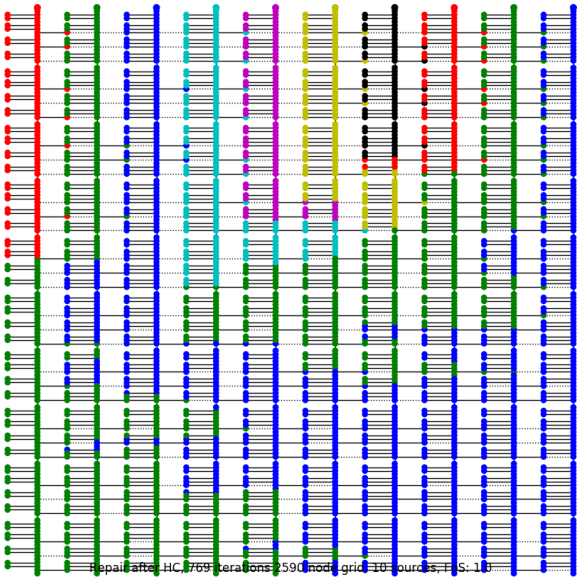


(b) Repair

Figure 22: The repair algorithm on the 4-source graph with 412 nodes. An edge failure occurs, causing loss of service in some nodes (in grey). Unserviced nodes are reassigned to sources with excess capacity.



(a) Link failure - disconnected nodes shown in grey



(b) Repair

Figure 23: The repair algorithm on the 10-source graph with 2590 nodes. An edge failure occurs, causing loss of service in some nodes (in grey). Unserviced nodes are reassigned to sources with excess capacity.

Chapter 7

A Flow-Based Heuristic Algorithm for Network Operations Planning in Smart Grids

Abstract

In this chapter, we present a heuristic algorithm for solving CSF based on computing the minimum-cost maximum flow on the graph. The algorithm outperforms the Local Search from Chapter 5 with respect to solution quality, and significantly with respect to running time. The work in this chapter has been published in (DFMV18).

7.1 Introduction

This work follows the same model, assumptions and problem as introduced in Section1.4, and Chapters 2, 5 and 6.

The work in this chapter is solely concerned with the Network Planning phase, i.e. the initial configuration of the network. However, the same algorithm can be used to handle reconfigurations of the network whenever a change in topology occurs, such as due to a fault, addition or removal of a source, or changes in demand.

7.2 Problem description and model

Following the model in (QCS14), we represent the electrical distribution grid as an undirected graph in which nodes represent energy sources, customers, or other operational elements (switches, circuit breakers and feeders), while edges represent arcs in the power network. Each source is assigned a capacity indicating the amount of commodity (electric power) that it can provide to the network, while each customer node has a demand in energy units. In the Network Planning phase, our objective is to connect each customer to a source, such that the capacity of the source is not exceeded by the combined demand of the customers it supplies. There are no capacity constraints on the lines (edges), only on the sources.

7.3 Algorithm

We present a heuristic algorithm for the solution of CSF that has two stages: an initialization stage and a balancing stage. In the initialization stage we obtain an initial solution by covering all customer vertices while permitting violations of the sources' capacities. In the subsequent balancing stage we improve the initial state by transferring excess vertices to other sources to correct the capacity violations. We determine the necessary node transfers by creating a flow network from the initial solution and solving the well-known Maximum Flow Minimum Cost problem on it. The resulting graph is a solution to CSF. The quality of the solution and the performance of the algorithm are discussed in Section 7.5.

The rationale behind this two-stage scheme is the following: Suppose that a leaf is connected to a source s_i by a single path. The only way to

cover this leaf is by building a tree rooted at source s_i. However, if the length of the path is close to the source's capacity c_i, the ratio of possible trees rooted in s_i covering the leaf out of all possible trees rooted in s_i will be small. Thus, a search algorithm will spend a lot of time finding any of the suitable trees. This is a case of *restrictive graph topology*: the more restrictive the topology is, the less feasible solutions exist. Such topological elements, such as the leaf in question, are very restrictive and search algorithms cannot efficiently find a solution unless they are guided to focus on restricted parts of a graph. However, it is hard to find an elegant and computationally efficient way to guide them when searching for trees of fixed weight or cardinality. In order to make this feasible, we violate capacity constraints in the first stage and assume that the cost of recovering them during the second stage will be small enough to let the algorithm outperform a search that does not violate the constraints.

7.3.1 Stage 1: Initialization

The initialization stage of the algorithm is similar to a Breadth-First Search with a priority queue. At each iteration of the algorithm we select the source with the largest capacity. This source then chooses a vertex from its neighboring vertices whose minimum distance (number of hops) to other sources is greatest. We then connect this vertex with the tree of the source.

Algorithm 5 Initialization algorithm

Let T_s be the tree of source s
repeat
  $s \leftarrow$ source with largest capacity
  $v \leftarrow$ neighbor of T_s with the largest minimum distance from any other source
  $T_s \leftarrow T_s \cup \{v\}$
until All vertices belong to a tree or no moves possible

As a result of the initialization algorithm run, we obtain a set of trees that cover the graph. Let x_i denote the size of T_i for all trees. These sizes

may be greater than the corresponding capacities $\{c_i\}$. The goal of the next stage is to change the obtained trees such that they still cover the graph and $c_i \geq x_i \ \ \forall i = \{1, \ldots, |S|\}$.

7.3.2 Stage 2: Improving the initial state

We create a flow graph $F = (S^+, T, f, c, a)$ in which vertices S^+ contain sources in the initial graph and two special vertices called the transfer sink and transfer source, and f, c and a are functions mapping flows, flow capacities and costs to all edges, respectively. The edge set T contains:

1. Edges between sources if it is possible to transfer vertices of the initial graph G between them (i.e. they have neighboring trees). Their flow capacity equals infinity and their cost equals 1.

2. Edges from the transfer source to the sources with a flow capacity of x_i (the size of T_i) and a cost of 0.

3. Edges from the sources to the transfer sink with a flow capacity of c_i (the capacity of the source) and cost equal to 0.

Using the flow graph F, we can now reduce the problem to the Minimum Cost Maximum Flow problem (SMW$^+$09), which is to find a maximum flow from a source vertex to a sink vertex that has minimum edge cost. In F, the flow between the transfer source and sources as well as flow between the sources and the transfer sink will be constant in every solution due to them having no cost at all. What is not known is how to organize a flow between the sources. The flow capacity between them is unlimited because there is no reason to limit the number of vertices that can be transferred from one source to another source. However, there is a cost on each edge limiting the number of such transfers between the sources, because each such transfer costs computational time.

Thus, a solution to the Maximum Flow problem is the optimal way to transfer all the vertices between the sources, with a minimum number of transfers being performed. In order to find it, we use the well-known

Ford-Fulkerson algorithm (AMO93; FF56). As a result, we obtain a number of vertices to be transferred between the sources. We then iterate on the edges and try to transfer vertices using the algorithm presented in the next subsection.

7.3.3 Algorithm for vertex transfering

Suppose we need to transfer t vertices from source s_1 to source s_2. Perform the following steps:

1. Count all cut vertices and their downstream children (v_c is a child of v_p if v_c would be disconnected from its source together with v_p) in T_1.
2. Transfer, with children, vertices from s_1 to s_2 that are neighboring T_2.

Perform steps 1 and 2 until we transfer t vertices or there are no vertices in T_1 that can be transferred to T_2.

Algorithm 6 Transfer algorithm

Require: t := number of vertices to be transfered per source
1: **repeat**
2: Count all cut vertices and their children (v_c is a child of v_p if v_c would be disconnected from its source together with v_p) in T_1.
3: Transfer (including children) vertices from T_1 to neighboring tree T_2.
4: **until** We have transferred t vertices or there are no vertices in T_1 that can be transferred to T_2.

At the end of the execution of this algorithm we should have a balanced capacitated spanning forest.

7.4 Experimental setup

To test the effectiveness of this algorithm, we performed experimental simulations of the algorithm compared to the Local Search heuristic from (DSV17)

and a priority queue BFS. Tests were performed on three types of graphs: 1) random trees with 1000 vertices and 20 sources, 2) random graphs with 1000 vertices and 1250 edges and 3) 32×32 square grids. We tested for cover quality, defined as $\frac{|V_{covered}|}{\min(\sum_{i \in S} c_i, |V|)}$, where $V_{covered}$ is the number of covered vertices, with respect to source capacity c_i and number of sources k. We also tested for time performance with respect to the number of sources and sources capacity. The results are summarized in Section 7.5.

7.5 Results

The results of our experiments are shown in Figures 24–32. In all Figures, the legend label "Flow" denotes our new algorithm as described in Section 7.3, compared to the state of the art "Local Search". The label "Flow2" refers to a variation of the algorithm in which the flow between vertices is set to a small value rather than infinity and the algorithm is run for several iterations (specified in the legend).

Our results demonstrate the superior performance in both cover quality and time performance of the new algorithm over a benchmark priority queue BFS and the Local Search from (DSV17).

For all experiments, each case was generated 50 times and the result was averaged. "Cover quality" is defined as the number of covered vertices divided by the minimum of sum of source capacities and number of all vertices.

7.5.1 Cover quality with respect to source capacity

Figures 24–26 show the superior coverage of the new algorithm compared to the Local Search and benchmark BFS with respect to varying source capacity, for different types of graphs: random trees, random graphs and square grids.

The cover quality reduces to around the value 50 because this is exactly $\frac{1000}{20}$, or $\frac{|V|}{k}$. This means a cover must be tight, or perfect, with all the total capacity being exhausted in order to cover all nodes, such that $|V| = \sum_{i \in S} c_i$. This state is difficult to achieve, as trees will block each other and lack reserve capacity to circumvent a block, leading to a reduced measure of cover quality. Conversely, if $|V| \neq \sum_{i \in S} c_i$, then $\frac{|V_{\text{covered}}|}{\min(|V|, \sum_{i \in S} c_i)}$ is expected to be greater, since trees can either cover a larger proportion of their capacity without blocking each other off, or they have reserve capacity to expand around a block. In either case, this leads to a better ratio of covered nodes.

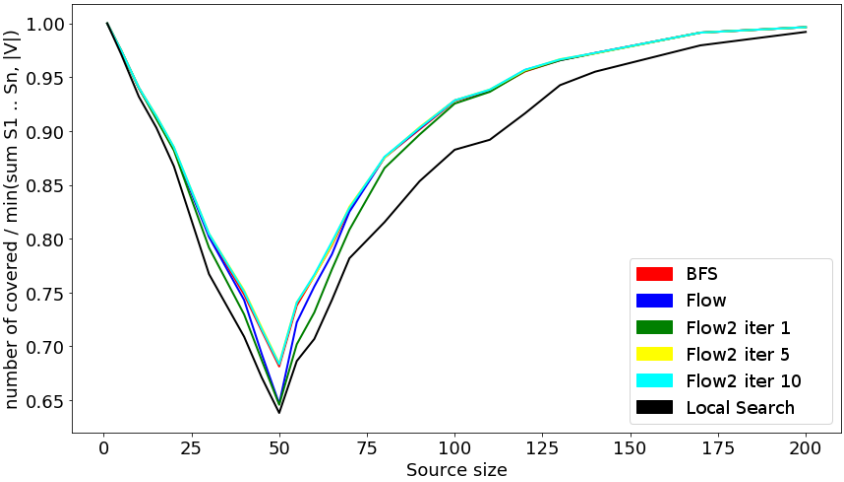


Figure 24: Cover quality with respect to source capacity on a random tree with 1000 vertices and 20 sources.

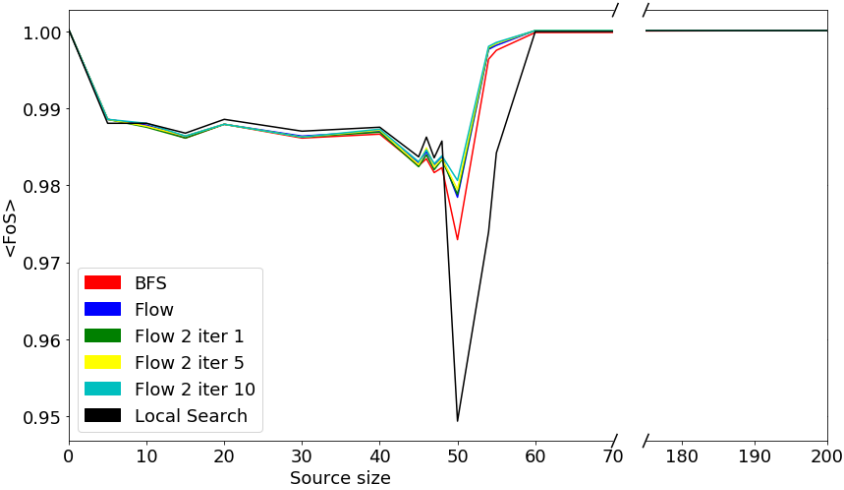


Figure 25: Cover quality with respect to source capacity on a random graph with 1000 vertices and 1250 edges.

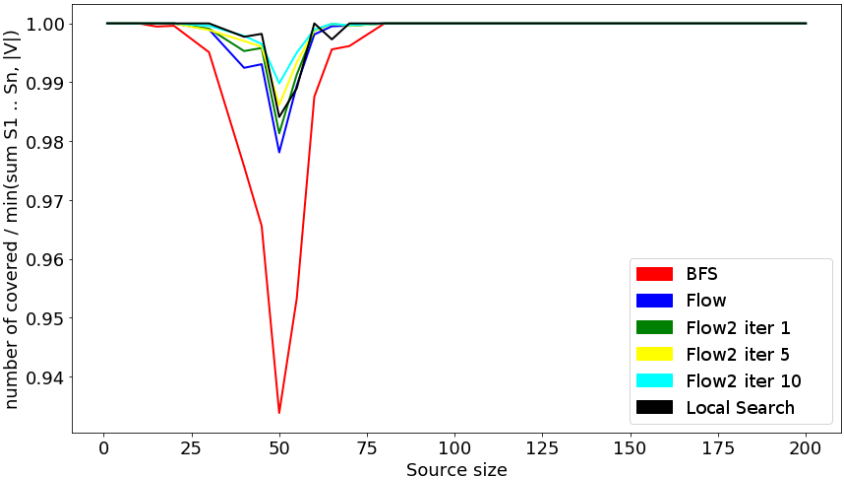


Figure 26: Cover quality with respect to source capacity on a 32×32 grid. Experiments were run as described in Fig. 24.

7.5.2 Cover quality with respect to number of sources

Figures 27–29 show the superior coverage of the new algorithm compared to Local Search with respect to the number of sources on graphs with roughly 1000 vertices. The difference in the cover quality between the three examples can be explained by the sparsity of the graph: In a tree, as we increase the number of sources they will begin to block each other. Random graphs and square grids are less sparse and provide more edges through which trees can avoid each other. In general, the more sources are added, cover quality decreased for all three graphs.

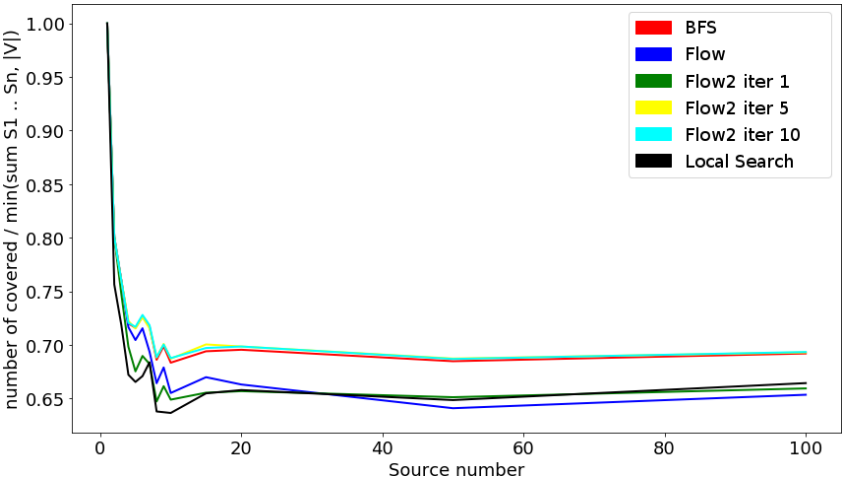


Figure 27: Cover quality w.r.t the number of sources on a random tree with 1000 nodes. Source size = $\frac{|V|}{k}$.

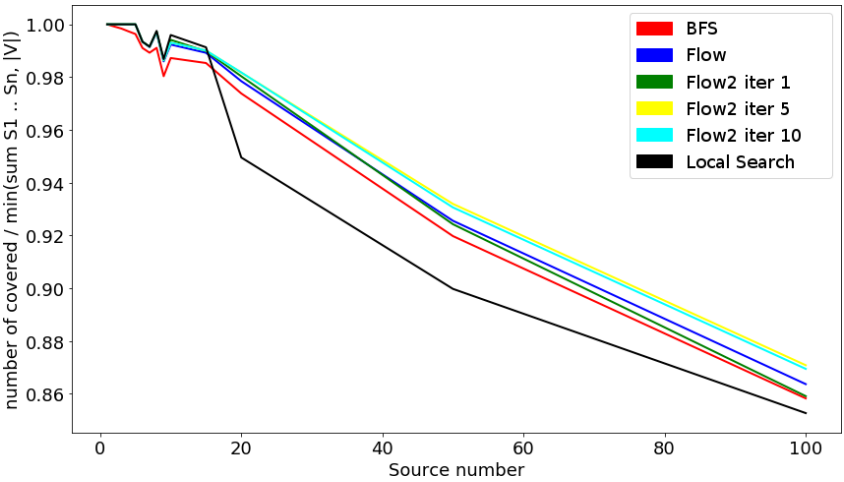


Figure 28: Cover quality w.r.t. the number of sources on a random graph with 1000 vertices and 1250 edges.

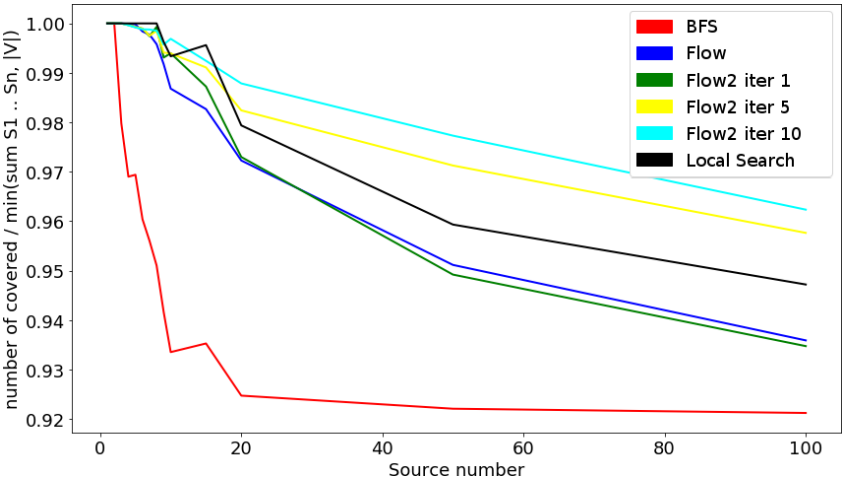


Figure 29: Cover quality w.r.t. the number of sources on a 32×32 grid.

7.5.3 Performance measures

Figure 30 shows the dramatically reduced wall time taken by the new algorithm in comparison with the Local Search, which is close to zero milliseconds. Figures 31–32 show the effect of varying the number of sources on elapsed time. When the number of vertices is high enough, the algorithm takes very little time to converge. However, when the number of vertices is smaller, the number of sources becomes more significant, as it gets more difficult to not obtain conflicts in the initial state. The growth of the number of sources thus causes the growth of the points in the flow graph, which increase time consumption. However, this effect can be observed only in the case when we have a high source–vertices ratio, which is not common in real-world situations.

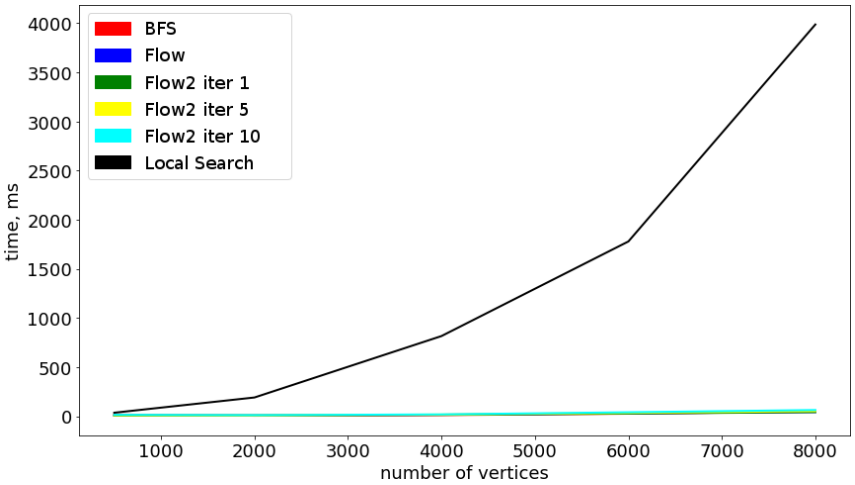


Figure 30: Elapsed wall time w.r.t. number of vertices (graph: tree with 20 sources)

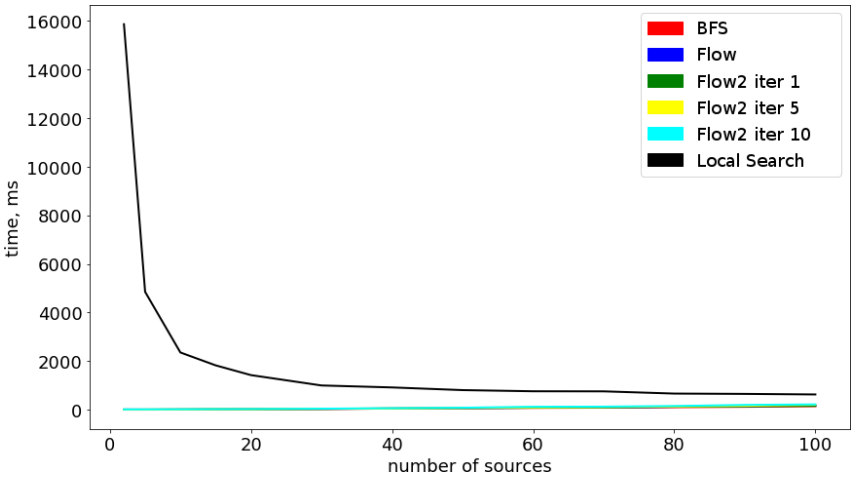


Figure 31: Elapsed wall time w.r.t. number of sources (graph: tree with 5000 vertices)

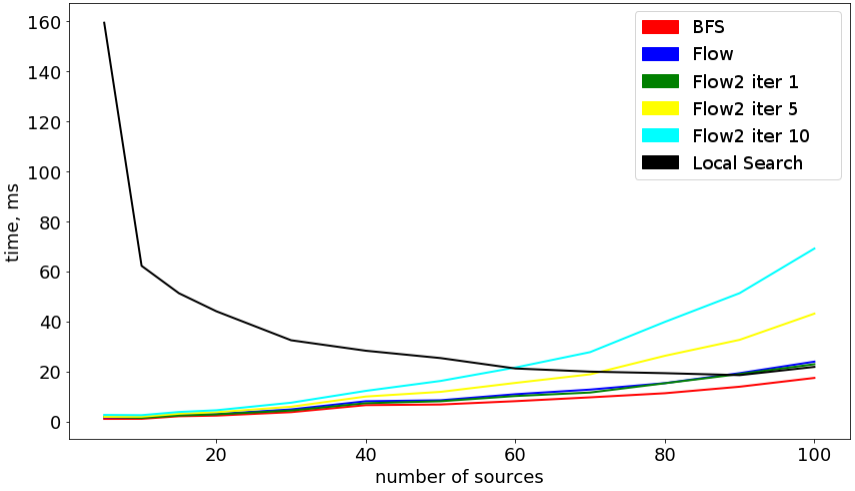


Figure 32: Elapsed wall time w.r.t. number of sources (graph: tree with 1000 vertices)

7.6 Conclusion

In this work we presented an algorithm for solving the Capacitated Spanning Forest problem with an application towards Network Operations Planning in smart grids. Our empirical findings indicate that this algorithm is more effective and efficient on a variety of graphs than previously reported algorithms for the same problem.

In future work we would like to test the algorithm also in repair scenarios and on real-world topologies. We also see that the algorithm performance can be improved by adding special policies on processing special cases (such as bridges or distant leaves) that are sources of invariants in the solutions. Early detection of such invariants will help to decrease the solution search time by decreasing the degrees of freedom of the search.

Finally, extensions towards more realistic features of the network, such as voltage constraints on lines, can help improve the applicability of the algorithm for studying more practical cases.

Chapter 8

Conclusion

In this thesis we addressed the problem of allocating multiple finite energy resources to customers in a smart grid in order to satisfy customer demands. We have also addressed the related problem of self-repair, namely reconnecting disconnected customers to energy sources. We have shown that these problems can be modeled as what we call the CAPACITATED SPANNING FOREST PROBLEM (CSF), which is the optimization problem of finding a vertex-disjoint spanning forest on a graph with a capacity constraint on each tree limiting its total weight.

Although several similar problems had been studied or at least been characterized in terms of their complexity in the literature, this problem did not have a substantial body of existing work addressing it. As such, we are the first to fully characterize the time complexity of this problem, although some special cases had been characterized in the literature. We proved the NP-completeness of the general problem for the cases with two sources, more than two sources, and sources with arbitrary capacities.

Although CSF is NP-complete, as with many NP-complete problems that do not admit polynomial-time solutions for the worst case, we can still attempt to develop heuristic solutions that work in the majority of cases that appear in practical applications. We provided three heuristic algorithms to address CSF in practice, which are the first algorithms in

the literature to do so. These are a Local Search heuristic, a Hill Climbing heuristic, and a Flow-based heuristic, each of which has been published in a peer-reviewed IEEE conference. Using a simulation-based approach we demonstrate the suitability of these algorithms for planning the initial configuration of a grid's topology. We have tested these algorithms on various types of graphs (square grids, trees and random graphs), as well as on graphs inspired by real-world grid topologies. We also demonstrated the application of the Local Search and Hill-Climbing heuristic to fault recovery. Our results show that heuristic approaches can successfully be applied to solving CSF in various scenarios.

8.1 Future work

Extending the work of (QCS14), we can test the resilience of different types of graphs that have not just one source but multiple sources while varying the number of failures in the network, and the heuristics used. One could investigate which graphs are the most resilient to failure, and what heuristics generate the most resilient structures at the Planning Phase, as well as provide the best recovery. An open problem would be to investigate what heuristics create the most resilient network structures at the network planning phase, and after repeated failures in the self-repair phase. Another interesting question would be to observe the effect that the sparsity of the graph has on the solution quality for each heuristic.

An important further goal of this work would be to perform simulations on the topology of a real-world distribution network, which would yield results directly relevant to a real scenario. Although we have used topologies *inspired* by real-world networks, we did not use real-world data itself. This data could be made available by an energy supplier from Brisbane, Australia. Hines et al. (HBSB10) also introduced an algorithm for generating graphs that have topological properties closely resembling real-world power networks. This can also lead to investigations on temporal networks, such that demands on the grid follow a certain schedule, and testing failures that occur according to those schedules.

In the field of designing practical algorithms, a standard open problem is to refine the existing algorithms, or design new algorithms that outperform the existing ones. In particular, it would be interesting to design an algorithm that takes the Minimum Spanning Tree during the initialization phase, rather than Breadth-First or Depth-First search. The authors note that we did try Simulated Annealing, however the results were unsatisfactory; this led, however to the design of the Hill Climbing heuristic. Of the existing algorithms in the literature, we can attempt to adapt Liang et al.'s (LSX13) approximation algorithm for CAPACITATED MINIMUM FOREST towards solving CSF.

Although in or work on CSF we have dealt with metaheuristics and heuristics, another approach that could be investigated is that of approximation algorithms. Approximation algorithms are polynomial-time algorithms for NP-complete problems that for all instances of the problem produce a solution whose value is within a factor α of the optimal solution. The main difference between approximation algorithms and heuristic approaches is that approximation algorithms are *guaranteed* to provide a solution within a factor α of the optimum, while heuristics are typically analyzed empirically. This performance guarantee α is demonstrated by a mathematical proof. Similarly, the time and space complexity of approximation algorithms are also proven mathematically. Thus, approximation algorithms can be said to be more mathematically rigorous than heuristic approaches, and are also used to provide a foundational initial solution which can be improved by heuristics. Not all hard problems admit approximation algorithms, and how well the solution to a problem can be approximated varies from problem to problem. Approximation algorithms have been found for many interesting NP-complete problems, but they can be challenging to find. The Christofides algorithm (Chr76) discovered in 1976 remains the best approximation algorithm for the famous TRAVELING SALESMAN problem, although algorithms with better approximation ratios have been discovered for special cases.

Another approach using mathematical programming, such as the Integer Programs described in Section 2.4 could also be investigated, and

will probably have to rely on a Branch and Bound heuristic using a Linear Programming relaxation in order to obtain a solution in a reasonable time.

Distributed algorithms are inherently interesting from a resilience perspective as a centralized architecture is more prone to attack or impact from unintended failures. The outage of a single top-level node will affect its client nodes lower in the hierarchy. By overcoming these limitations of centrality, a distributed protocol intrinsically introduces greater robustness and resilience, and likewise networks based on distributed routing are naturally able to bootstrap. Thus, algorithms that construct a solution from a bottom-up rather than top-down perspective are another area of investigation. The Local Search algorithm we have introduced fulfils this criteria and could be implemented in a distributed fashion.

In its present state, the model ignores the dynamic electrical properties of the grid such as the magnitude of electrical flows, capacities of components, and consumer demand. Thus, mere topology-based routing could in fact disable more lines or substations by overloading them, thereby hindering network recovery or disabling more of the grid through a cascading failure. Awareness of flows and components' capacities is important not just for isolated failures but also in the incidence of blackouts, when a large portion of the grid shuts down and needs to be restarted in sequence according to these properties. In the future, the algorithms can take into consideration properties beyond just the network's physical topology, such as limits on lines' carrying-flow capacity, electric path resistance, pricing of electricity, preference of some sources of energy over others (such as favoring renewable sources), etc., in order to not just restore connectivity, but also do so optimally with respect to metrics such as load distribution and the preferences of each consumer. Additional evaluation metrics besides $\langle \text{FoS} \rangle$ include the time t of undelivered power to each load or for the whole grid after a failure, the difference between the power consumed by a component and its maximum capacity, the local or grid-wide power loss due to resistance, and statistical measures on the graph such as node-connectivity.

There are many further possible extensions that were outside the scope

of the current work. These include time-dependent data such as modeling variable demands and demand response, or modelling intermittent energy generations.

References

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, April 1987. 39

[AK05] Sanjeev Arora and George Karakostas. A 2 + approximation algorithm for the k-MST problem. *Mathematical Programming*, 107(3):491–504, December 2005. 44

[ALS88] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Problems easy for tree-decomposable graphs. In Timo Lepist o and Arto Salomaa, editors, *Automata, Languages and Programming*, number 317 in Lecture Notes in Computer Science, pages 38–51. Springer Berlin Heidelberg, July 1988. DOI: 10.1007/3-540-19488-6_105. 39

[ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, June 1991. 39

[AMO93] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993. 88

[BH06] A. Björklund and T. Husfeldt. Inclusion–Exclusion Algorithms for Counting Set Partitions. In *47th Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS '06*, pages 575–582, October 2006. 38

[BHK09] A. Björklund, T. Husfeldt, and M. Koivisto. Set Partitioning via Inclusion–Exclusion. *SIAM Journal on Computing*, 39(2):546–563, January 2009. 38, 44

[Bod86] Hans Leo Bodlaender. *Classes of graphs with bounded tree-width*. Department of Computer Science, University of Utrecht, 1986. 39

[Bod88] Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In Timo Lepist and Arto Salomaa, editors, *Automata, Languages and Programming*, number 317 in Lecture Notes in Computer Science, pages 105–118. Springer Berlin Heidelberg, July 1988. DOI: 10.1007/3-540-19488-6_110. 39, 43

[BPT92] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(1-6):555–581, June 1992. 44

[CDH+10] Guo Chen, Zhao Yang Dong, David J. Hill, Guo Hua Zhang, and Ke Qian Hua. Attack structural vulnerability of power grids: A hybrid approach based on complex networks. *Physica A: Statistical Mechanics and its Applications*, 389(3):595–603, feb 2010. 55

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976. 99

[CI93] Bahman Kalantari Celina Imieliska. A greedy heuristic for a minimum-weight forest problem. *Operations Research Letters*, (2):65–71, 1993. 41

[CKH95] Pierluigi Crescenzi, Viggo Kann, and Magnús Halldórsson. A compendium of np optimization problems, 1995. 43

[CM04] Roberto Cordone and Francesco Maffioli. On the complexity of graph tree partition problems. *Discrete Applied Mathematics*, 134(13):51–65, January 2004. 33

[Col10] Steven Collier. Ten Steps to a Smarter Grid. *IEEE Industry Applications Magazine*, 16(2):62–68, March 2010. 1

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971. 46, 47

[Cou90] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, March 1990. 39

[CPP+15] M. Coppo, P. Pelacchi, F. Pilo, G. Pisano, G.G. Soma, and R. Turri. The Italian smart grid pilot projects: Selection and assessment of the test beds for the regulation of smart electricity distribution. *Electric Power Systems Research*, 120:136–149, March 2015. 3

[Dav18] George Davidescu. The Capacitated Spanning Forest Problem is NP-complete. 9th International Conference on Complex Systems, July 2018. 46

[DCI13] Felicita Di Giandomenico, Silvano Chiaradonna, and Davide Iacono. Smart Control of Energy Distribution Grids over Heterogeneous Communication Networks. Technical report, Consiglio Nazionale delle Ricerche, 2013. 5

[DFFM15] Maurizio Delfanti, Davide Falabretti, Massimo Fiori, and Marco Merlo. Smart Grid on field application in the Italian framework: The A.S.SE.M. project. *Electric Power Systems Research*, 120:56–69, March 2015. 3

[DFMV18] © 2018 IEEE. Reprinted, with permission, from G. Davidescu, A. Filchenkov, A. Muratov, and V. Vyatkin. A flow-based heuristic algorithm for network operations planning in smart grids. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, pages 3529–3534, Oct 2018. v, 84

[DFOP15] Maurizio Delfanti, Enrico Fasciolo, Valeria Olivieri, and Mauro Pozzi. A2a project: A practical implementation of smart grids in the urban area of Milan. *Electric Power Systems Research*, 120:2–19, March 2015. 3

[DSV17] © 2017 IEEE. Reprinted, with permission, from G. Davidescu, T. Stützle, and V. Vyatkin. Network planning in smart grids via a local search heuristic for spanning forest problems. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pages 1212–1218, June 2017. v, xiv, 15, 21, 30, 52, 53, 70, 71, 72, 74, 75, 76, 77, 80, 88, 89

[DV17] © 2017 IEEE. Reprinted, with permission, from G. Davidescu and V. Vyatkin. Network planning and self-repair in models of urban distribution networks via hill climbing. In *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 5477–5482, Oct 2017. v, 30, 52, 69

[EGK+03a] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Covering Graphs Using Trees and Stars. In Sanjeev Arora, Klaus Jansen, Jos D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, number 2764 in Lecture Notes in Computer Science, pages 24–35. Springer Berlin Heidelberg, 2003. 42

[EGK+03b] G. Even, N. Garg, J. Knemann, R. Ravi, and A. Sinha. Covering Graphs Using Trees and Stars. In Sanjeev Arora, Klaus Jansen, Jos D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, number 2764 in Lecture Notes in Computer Science, pages 24–35. Springer Berlin Heidelberg, 2003. 42

[Exe13] Executive Office of the President, Council of Economic Advisers. *Economic Benefits of Increasing Electric Grid Resilience to Weather Outages*. The Council, 2013. 1

[Fel11] Andreas Emil Feldmann. Fast Balanced Partitioning is Hard, Even on Grids and Trees. *arXiv:1111.6745 [cs]*, November 2011. arXiv: 1111.6745. 43

[FF56] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956. 88

[FMXY12] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart Grid The New and Improved Power Grid: A Survey. *IEEE Communications Surveys & Tutorials*, 14(4):944–980, January 2012. 10, 11, 13

[FP14] Gianni Fenu and Pier Luigi Pau. A Model of Assessment of Collateral Damage on Power Grids based on Complex Network Theory. *Procedia Computer Science*, 32:437–444, 2014. 12

[Gar05] Naveen Garg. Saving an Epsilon: A 2-approximation for the k-MST Problem in Graphs. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 396–402, New York, NY, USA, 2005. ACM. 44

[GBH97] Nili Guttmann-Beck and Refael Hassin. Approximation Algorithms for MinMax Tree Partition. *Journal of Algorithms*, 24(2):266–286, August 1997. 34, 39, 40

[GBH98] Nili Guttmann-Beck and Refael Hassin. Approximation algorithms for minimum tree partition. *Discrete Applied Mathematics*, 87(13):117–137, October 1998. 40

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. 38, 43, 47

[GW95] M. Goemans and D. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24(2):296–317, April 1995. 32, 41, 43

[GWP+14] Christopher Greer, David A. Wollman, Dean E. Prochaska, Paul A. Boynton, Jeffrey A. Mazer, Cuong T. Nguyen, Gerald J. FitzPatrick, Thomas L. Nelson, Galen H. Koepke, Allen R. Hefner Jr, Victoria Y. Pillitteri, Tanya L. Brewer, Nada T. Golmie, David H. Su, Allan C. Eustis, David G. Holmberg, and Steven T. Bushby. NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0. Technical Report NIST SP 1108r3, National Institute of Standards and Technology, October 2014. 1, 2, 7, 9, 14

[HBFD+10] P Hoffman, W Bryan, M Farber-DeAnda, M Cleaver, C Lewandowski, and K Young. Hardening and resiliency, us energy industry response to recent hurricane seasons. *Office of Electricity Delivery and Energy Reliability, US Department of Energy, OE/ISER Final Report*, 2010. 13

[HBSB10] P. Hines, S. Blumsack, E. Cotilla Sanchez, and C. Barrows. The Topological and Electrical Structure of Power Grids. In *2010 43rd Hawaii International Conference on System Sciences*, pages 1–10. IEEE, 2010. 55, 98

[HRW16] F. Hong, L. Ru, and L. Weiyang. Distribution network planning method with efficiency power plant. In *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pages 2521–2526, June 2016. 15

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008. 23, 59, 74

[HVNS11] N. Higgins, V. Vyatkin, N. C. Nair, and K. Schwarz. Distributed power system automation with iec 61850, iec 61499, and intelligent control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(1):81–92, Jan 2011. 70, 75

[HWR+13] Zhen Huang, Cheng Wang, Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. Modeling cascading failures in smart power grid using interdependent complex networks and percolation theory. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pages 1023–1028. IEEE, jun 2013. 55

[IKSW13] Swami Iyer, Timothy Killingback, Bala Sundaram, and Zhen Wang. Attack Robustness and Centrality of Complex Networks. *PLoS ONE*, 8(4):e59613, jan 2013. 55

[Jai15] S.R. Jaiswal. Optimized Technique for Capacitated Minimum Forest Problem In Wireless Sensor Networks. In *2015 International Conference on Emerging Information Technology and Engineering Solutions (EITES)*, pages 13–17, February 2015. 32

[JS15] S.R. Jaiswal and R.R. Sawant. Capacitated minimum forest problem in wireless sensor networks. In *2015 International Conference on Communication, Information Computing Technology (ICCICT)*, pages 1–5, January 2015. 32

[KBD+12] Tiffany Hyun-Jin Kim, K. Brancik, D. Dickinson, A. Perrig, and B. Sinopoli. CyberPhysical Security of a Smart Grid Infrastructure. *Proceedings of the IEEE*, 100(1):195–209, January 2012. x, 2, 13, 14

[KHGP13] Ruiyuan Kong, Congying Han, Tiande Guo, and Wei Pei. An Energy-Based Centrality for Electrical Networks. *Energy and Power Engineering*, 05(04):597–602, 2013. 55

[KS11] M. Reza Khani and Mohammad R. Salavatipour. Improved Approximation Algorithms for the Min-Max Tree Cover and Bounded Tree Cover Problems. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and Jos D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, number 6845 in Lecture Notes in Computer Science, pages 302–314. Springer Berlin Heidelberg, 2011. 42

[LDS10] Gang Lu, D. De, and Wen-Zhan Song. Smartgridlab: A laboratory-based smart grid testbed. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 143–148, Oct 2010. 1, 13, 14

[LM05] Michael Laszlo and Sumitra Mukherjee. Another greedy heuristic for the constrained forest problem. *Operations Research Letters*, 33(6):629–633, November 2005. 41

[LM06] Michael Laszlo and Sumitra Mukherjee. A class of heuristics for the constrained forest problem. *Discrete Applied Mathematics*, 154(1):6–14, January 2006. 41

[LM11] Michael Laszlo and Sumitra Mukherjee. A Genetic Algorithm for the Constrained Forest Problem. 2011. 41

[LSX13] Weifa Liang, P. Schweitzer, and Zichuan Xu. Approximation Algorithms for Capacitated Minimum Forest Problems in Wireless Sensor Networks with a Mobile Sink. *IEEE Transactions on Computers*, 62(10):1932–1944, October 2013. 31, 33, 99

[LZ09] Chung-Shou Liao and Louxin Zhang. Approximating the Spanning k-Tree Forest Problem. In Xiaotie Deng, John E. Hopcroft, and Jinyun Xue, editors, *Frontiers in Algorithmics*, number 5598 in Lecture Notes in Computer Science, pages 293–301. Springer Berlin Heidelberg, 2009. 44

[OK01] Julie Osborn and Cornelia Kawann. Reliability of the us electricity system: Recent trends and current issues. 2001. 13

[PA13] G.A. Pagani and M. Aiello. Modeling the last mile of the smart grid. In *Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES*, pages 1–6, Feb 2013. 21

[Pap78a] C. H. Papadimitriou. The complexity of the capacitated tree problem. *Networks*, 8(3):217–230, 1978. 31, 36, 37

[Pap78b] Christos H. Papadimitriou. The complexity of the capacitated tree problem. *Networks*, 8(3):217–230, 1978. 48

[PPL+16] Francesca Possemato, Maurizio Paschero, Lorenzo Livi, Antonello Rizzi, and Alireza Sadeghian. On the impact of topological properties of smart grids in power losses optimization problems. *International Journal of Electrical Power & Energy Systems*, 78:755–764, June 2016. 4

[QCS14] Walter Quattrociocchi, Guido Caldarelli, and Antonio Scala. Self-healing networks: redundancy and structure. *PloS One*, 9(2):e87986, January 2014. xi, 19, 20, 21, 22, 54, 55, 71, 85, 98

[Ros95] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995. 59, 74

[Ros10] Michael G. Rosenfield. The smart grid and key research technical challenges. In *2010 Symposium on VLSI Technology*, pages 3–8. IEEE, June 2010. 1, 2, 4

[RP14] Sushmita Ruj and Arindam Pal. Analyzing Cascading Failures in Smart Grids under Random and Targeted Attacks. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 226–233. IEEE, May 2014. x, 15, 16

[RSM+96] R. Ravi, R. Sundaram, M. Marathe, D. Rosenkrantz, and S. Ravi. Spanning TreesShort or Small. *SIAM Journal on Discrete Mathematics*, 9(2):178–200, May 1996. 44

[SBP+08] Ingve Simonsen, Lubos Buzna, Karsten Peters, Stefan Bornholdt, and Dirk Helbing. Dynamic Effects Increasing Network Vulnerability to Cascading Failures. *Physical Review Letters*, 100(21), may 2008. 55

[SCC+14] Antonio Scala, Guido Caldarelli, Alessandro Chessa, Alfonso Damiano, Mario Mureddu, Sakshi Pahwa, Caterina Scoglio, and Walter Quattrociocchi. Power Grids, Smart Grids and Complex Networks. In Davron Matrasulov and H. Eugene Stanley, editors, *Nonlinear Phenomena in Complex Systems: From Nano to Macro Scale*, NATO Science for Peace and Security Series C: Environmental Security, pages 97–110. Springer Netherlands, January 2014. x, 5, 12

[SMC+13] Antonio Scala, Mario Mureddu, Alessandro Chessa, Guido Caldarelli, and Alfonso Damiano. Distributed Generation and Resilience in Power Grids. *Critical Information Infrastructures Security*, pages 71–79, January 2013. 21

[SMRP11] Nuno Silva, David Marsh, Alberto Rodrigues, and Carlos MOTA Pinto. Dynamic SCADA/DMS Data Model - Plug & Play Smart Grid Solutions. (1022):5, 2011. 8

[SMW+09] Xinyu Shen, Limin Meng, Yifan Wu, Kai Zhou, and Yuhua Zhang. Research on energy-aware routing protocol based on min cost max flow algorithm for mobile ad hoc networks. In *Computational Intelligence and Design, 2009. ISCID'09. Second International Symposium on*, volume 2, pages 30–33. IEEE, 2009. 87

[SXY+16] Y. Shen, J. Xu, Y. Yue, S. Zhu, and Q. Li. Comprehensive coordinated model of active distribution network planning. In *2016 IEEE International Conference on Power System Technology (POWERCON)*, pages 1–6, September 2016. 15, 54

[TA16] Maria Lorena Tuballa and Michael Lochinvar Abundo. A review of the development of Smart Grid technologies. *Renewable and Sustainable Energy Reviews*, 59:710–725, June 2016. 3, 4

[YTK96] Takeo Yamada, Hideo Takahashi, and Seiji Kataoka. A heuristic algorithm for the mini-max spanning forest problem. *European Journal of Operational Research*, 91(3):565–572, June 1996. 34, 41

[ZVD15] Gulnara Zhabelova, Valeriy Vyatkin, and Victor N Dubinin. Toward Industrially Usable Agent Technology for Smart Grid Automation. *IEEE Transactions on Industrial Electronics*, 62(4):2629–2641, 2015. x, 16, 17, 70