IMT Institute for Advanced Studies, Lucca
Lucca, Italy

Auctions and Mechanisms in Keyword-based Advertising

PhD Program in Computer Science and Engineering
XXIII Cycle

By
Michele Budinich

2011

The dissertation of Michele Budinich is approved.

Programme Coordinator: Ugo Montanari, Università di Pisa, Italy

Supervisor: Bruno Codenotti, IIT - CNR Pisa, Italy

Tutor: Marzia Buscemi, IMT Lucca, Italy

The dissertation of Michele Budinich has been reviewed by:

Elias Koutsoupias, University of Athens, Greece

Giuseppe Persiano, Università di Salerno, Italy

IMT Institute for Advanced Studies, Lucca

2011

Table of Contents

Acknowledgments

Part of the work presented in this thesis was co-authored with: Bruno Codenotti, Lance Fortnow, Filippo Geraci, Marco Pellegrini and Rakesh Vohra. The relevant publications are:

- Budinich and Fortnow [2011], published in the Proceedings of the 11th ACM Conference on Electronic Commerce (EC).
- Budinich et al. [2010a], published in the Proceedings of the Web-EC Conference.
- Budinich et al. [2011], published in the Proceedings of the IEEE CCSIE Conference.
- Budinich et al. [2010b], which is an ongoing research project and has not yet been submitted for publication.

Vita

11 February, 1982	Born, Trieste, Italy
2003-2004	Semester as a visiting student (Erasmus Programme) Universiteit Leiden Leiden, Netherlands
July 2004	Bachelor Degree in Computer Science Università Ca′ Foscari Venezia, Italy
February 2007	Master of Science Degree in Computer Science Università di Pisa Pisa, Italy
2007	Research Grant IIT - CNR Pisa Pisa, Italy
2010 - 2011	Three semesters as a visiting PhD Student Northwestern University Evanston, U.S.A.
December 2011	PhD in Computer Science and Engineering IMT - Institute for Advanced Studies Lucca, Italy

Publications

M. Budinich. *Quality Scores in Sponsored Search*. Technical Report, IIT - CNR Pisa, 2011.

M. Budinich and L. Fortnow. *Repeated Matching Pennies with Limited Randomness*. ACM EC, 2011.

M. Budinich, B. Codenotti, F. Geraci and M. Pellegrini. *Estimating Click Through Rates for Adwords using Public Data*. IEEE CCSIE, 2011.

M. Budinich, B. Codenotti, F. Geraci and M. Pellegrini. *On the Benefits of Keyword Spreading in Sponsored Search Auctions*. EC-Web, 2010.

Abstract

Today most of search engines' profits come from advertising, and in particular from sponsored search. In sponsored search, advertisement slots next to search results are sold. When a query is made, besides processing the query results themselves, the search engine selects ads relevant to that query. Two of the main characteristics of this form of advertising are that advertisers are billed only when a click on their ad is made, and that prices are computed using an auction.

In this thesis we consider some generalizations of the sponsored search auction model presented in the literature. In particular we account for the fact that the search engines have a great control over the order in which advertisers are ranked. In fact search engines assign quality scores to each advertiser, and, prior to sorting, scale all bids by such factors. We show how this changes the main properties of the equilibria in these auctions, and that in particular, the efficiency directly depends on how the search engine sets the quality scores. We then analyze some strategic behaviors in this environment, showing how their properties can differ from other classical auction models.

In a second part of this thesis we present some experimental results. These were obtained with a large scale sponsored search simulator developed for this thesis. To realistically simulate such environments new algorithms and techniques were developed that partially overcome the lack of publicly available information, by effectively estimating many hidden parameters, such as click through rates. The experimental results presented focus on the global effects on the market when a fraction of the advertising agents engage in strategic behaviors. In particular we focus on two cases. In the first agents start to optimize their set of keywords using a custom build set of available synonyms. The second behavior we consider is one in which an agent changes his bid, with the objective of increasing costs for his opponents. Interestingly we show that this technique is not always profitable.

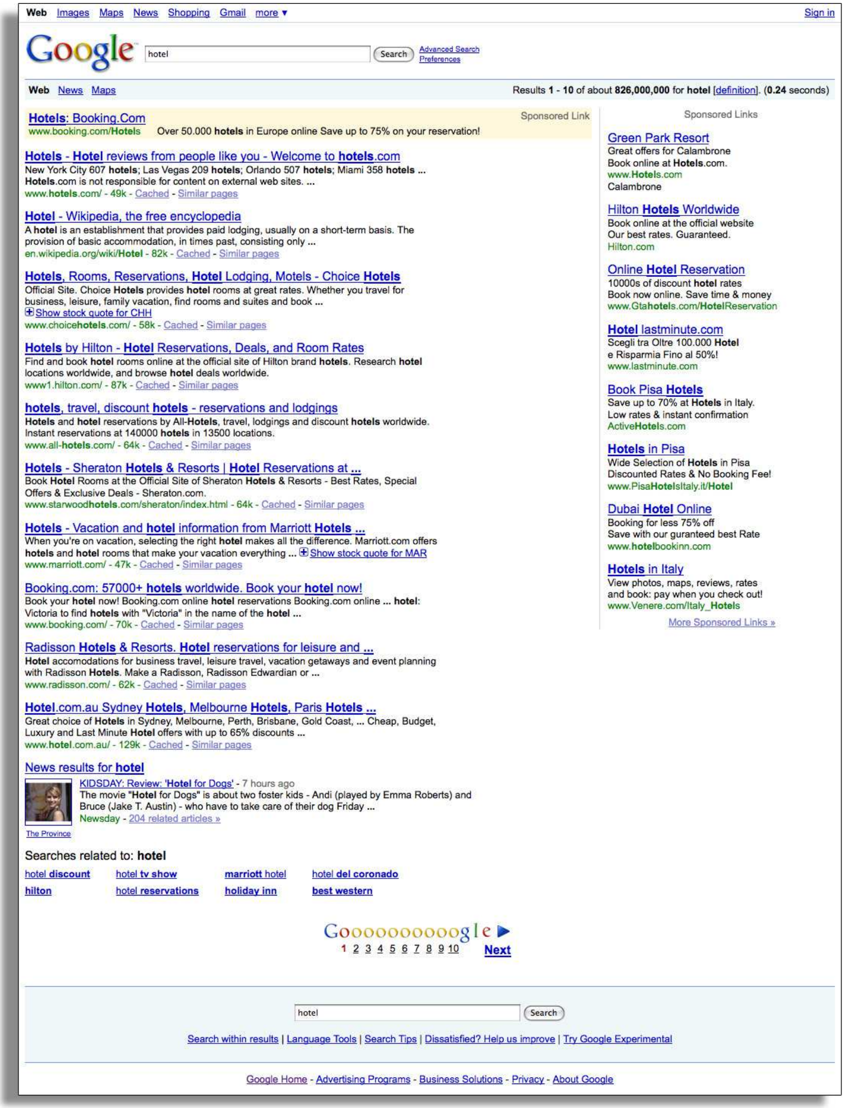
The fact that, in sponsored search, agents might not have access to all the necessary information to compute their optimal bids, marks a significant departure from the theoretical models, that instead assume full knowledge of the environment. The field that studies models in which agents' capabilities are limited is called bounded rationality. The final chapters of this thesis are dedicated to work done by the author during his visiting period at Northwestern University, Evanston U.S.A., under the supervision of prof. Lance Fortnow. The focus is on bounded rationality, and two problem that directly expose the role of computational limitations in game theory are analyzed.

Introduction

This thesis is about online ad auctions, which are the mechanisms used by most search engines to decide which sponsored links to display along search results (see Figure 1). The number of available slots in which ads can be displayed is usually limited, while the number of interested customers can be very large. Search engines assign slots to bidders by mean of an auction. The online ad auction problem is primarily concerned with the design and analysis of such auctions. It is an example of a new kind of problems, lying at the intersection between computer science, game theory, and economic theory.

There are many other interesting problems at the border between these disciplines, and this research area has appropriately been called algorithmic game theory. Probably the key driving force behind the development of this new area has been the emergence of new computational artifacts, most notably the Internet. One of the main characteristics of the Internet is that it is made up of and operated by a number of diverse agents, each with different interests and objectives. The classical tools of computer science are inadequate to analyze and fully understand these new scenarios, and the computer science community turned to research fields where the interaction of many agents with different interests is central: game theory and economics.

This introduction gives a very brief outline of the main goals of game theory and economic theory, and their applications in computer science, followed by an overview of this thesis: from the related literature to the results presented.



Results 1 - 10 of about 826,000,000 for hotel [definition]. (0.24 seconds)

Sponsored Link

Hotels: Booking.Com
www.booking.com/Hotels Over 50.000 hotels in Europe online Save up to 75% on your reservation!

Hotels - Hotel reviews from people like you - Welcome to hotels.com
New York City 607 hotels; Las Vegas 209 hotels; Orlando 507 hotels; Miami 358 hotels ... Hotels.com is not responsible for content on external web sites. ...
www.hotels.com/ - 49k - Cached - Similar pages

Hotel - Wikipedia, the free encyclopedia
A hotel is an establishment that provides paid lodging, usually on a short-term basis. The provision of basic accommodation, in times past, consisting only ...
en.wikipedia.org/wiki/Hotel - 82k - Cached - Similar pages

Hotels, Rooms, Reservations, Hotel Lodging, Motels - Choice Hotels
Official Site. Choice Hotels provides hotel rooms at great rates. Whether you travel for business, leisure, family vacation, find rooms and suites and book ...
⊞ Show stock quote for CHH
www.choicehotels.com/ - 58k - Cached - Similar pages

Hotels by Hilton - Hotel Reservations, Deals, and Room Rates
Find and book hotel rooms online at the official site of Hilton brand hotels. Research hotel locations worldwide, and browse hotel deals worldwide.
www1.hilton.com/ - 87k - Cached - Similar pages

hotels, travel, discount hotels - reservations and lodgings
Hotels and hotel reservations by All-Hotels, travel, lodgings and discount hotels worldwide. Instant reservations at 140000 hotels in 13500 locations.
www.all-hotels.com/ - 64k - Cached - Similar pages

Hotels - Sheraton Hotels & Resorts | Hotel Reservations at ...
Book Hotel Rooms at the Official Site of Sheraton Hotels & Resorts - Best Rates, Special Offers & Exclusive Deals - Sheraton.com.
www.starwoodhotels.com/sheraton/index.html - 64k - Cached - Similar pages

Hotels - Vacation and hotel information from Marriott Hotels ...
When you're on vacation, selecting the right hotel makes all the difference. Marriott.com offers hotels and hotel rooms that make your vacation everything ... ⊞ Show stock quote for MAR
www.marriott.com/ - 47k - Cached - Similar pages

Booking.com: 57000+ hotels worldwide. Book your hotel now!
Book your hotel now! Booking.com online hotel reservations Booking.com online ... hotel: Victoria to find hotels with "Victoria" in the name of the hotel ...
www.booking.com/ - 70k - Cached - Similar pages

Radisson Hotels & Resorts. Hotel reservations for leisure and ...
Hotel accomodations for business travel, leisure travel, vacation getaways and event planning with Radisson Hotels. Make a Radisson, Radisson Edwardian or ...
www.radisson.com/ - 62k - Cached - Similar pages

Hotel.com.au Sydney Hotels, Melbourne Hotels, Paris Hotels ...
Great choice of Hotels in Sydney, Melbourne, Perth, Brisbane, Gold Coast, ... Cheap, Budget, Luxury and Last Minute Hotel offers with up to 65% discounts ...
www.hotel.com.au/ - 129k - Cached - Similar pages

News results for hotel

KIDSDAY: Review: 'Hotel for Dogs' - 7 hours ago
The movie "Hotel for Dogs" is about two foster kids - Andi (played by Emma Roberts) and Bruce (Jake T. Austin) - who have to take care of their dog Friday ...
Newsday - 204 related articles »

The Province

Searches related to: hotel

hotel discount
hotel tv show
marriott hotel
hotel del coronado
hilton
hotel reservations
holiday inn
best western

Sponsored Links

Green Park Resort
Great offers for Calambrone
Book online at Hotels.com.
www.Hotels.com
Calambrone

Hilton Hotels Worldwide
Book online at the official website
Our best rates. Guaranteed.
Hilton.com

Online Hotel Reservation
10000s of discount hotel rates
Book now online. Save time & money
www.Gtahotels.com/HotelReservation

Hotel lastminute.com
Scegli tra Oltre 100.000 Hotel
e Risparmia Fino al 50%!
www.lastminute.com

Book Pisa Hotels
Save up to 70% at Hotels in Italy.
Low rates & instant confirmation
ActiveHotels.com

Hotels in Pisa
Wide Selection of Hotels in Pisa
Discounted Rates & No Booking Fee!
www.PisaHotelsItaly.it/Hotel

Dubai Hotel Online
Booking for less 75% off
Save with our guranteed best Rate
www.hotelbookinn.com

Hotels in Italy
View photos, maps, reviews, rates
and book: pay when you check out!
www.Venere.com/Italy_Hotels

More Sponsored Links »

Gooooooooooogle ▶
1 2 3 4 5 6 7 8 9 10 Next
hotel
Search
Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve | Try Google Experimental
Google Home - Advertising Programs - Business Solutions - Privacy - About Google

Figure 1: Example of search engine results page, with sponsored links visible at the top and to the right of the search results.

Game Theory

Game theory was founded by von Neumann and Morgenstern[1] as a general theory of interaction between decision-makers. In this sense a game can be thought of as a description of strategic interaction. The fundamental assumption of game theory is that all players are rational, meaning they pursue some well-defined objectives, and that they all reason strategically, that is they take into account their knowledge (or expectations) of other players' behavior.

There are many different kinds of games, intended in this broad sense. One of the first and most basic models are the strategic games, or games in normal form. In these games each participant is defined by the set of possible actions he can take and by a preference relation over the set of all possible actions by all other players.

A solution to a game is a systematic description of the outcomes that may emerge depending on all of the players' actions. The most widely used and successful solution concept in game theory is the Nash equilibrium. This notion captures the essence of stability in this context. Assume each player is assigned a strategy: this set of strategies constitutes a Nash equilibrium if no player has an action yielding an outcome that he prefers, given the actions assigned to all other players. Informally it is the best each player can do unilaterally. In 1951 Nash[2] proved that if the set of actions for each player is convex and his preferences are continuous and quasi-concave then a Nash equilibrium always exists. An interesting case is when we allow players to choose one of their strategies randomly. That is each player has his set of strategies and a probability distribution over such a set, describing how likely it is that he will play that particular strategy. The sets of strategies equipped with a probability distribution are called mixed strategies. Given a finite set of strategies, the corresponding set of mixed strategies is always convex, and so it follows directly from Nash's theorem that every finite strategic game has a Nash equilibrium in the mixed strategies.

The extremely broad definition of strategic game and the comparatively weak assumptions needed, make Nash's theorem a fundamental result.

[1] J. von Neumann, O. Morgenstern *Theory of Games and Economic Behavior*, Princeton University Press, 1944

[2] J.F. Nash, *Non-Cooperative Games*, Annals of Mathematics (**54**), 1951

Microeconomic Theory

In economic theory, and particularly in microeconomic theory, the central topic is the study of the interactions between diverse agents whose actions are determined by suitable economic incentives. The main difference with respect to game theory is the presence of prices, which can be seen as an efficient and ingenious mean of decentralization.

An easy to describe of market is given by exchange economies, where the production of goods is not considered. There are a number of agents, each of which is in possession of a non-negative vector of goods: his initial endowment. Furthermore each agent has a utility function that, given any two baskets of goods, assigns a higher value to the one preferred by the agent. It could very well be that two (or more) agents are dissatisfied with their initial endowments, and would like to give some goods away and receive others such as to obtain a higher utility.

When each good is assigned a price, the rational behavior for each agent is to sell his initial endowment, and use the money he receives to find a bundle of goods that maximizes his utility. The constraints to this maximization problem are given through prices themselves, and state that one cannot spend more than he initially holds (i.e. the value of his initial endowment at current prices); these are the budget constraints.

Since we are not considering production we can obtain the total supply of goods in the market simply by summing up all the initial endowments. The demand of the market, at any given prices, is the sum over all agents of the bundles that solve each agent's maximization problem. In other words, having fixed prices, we let each agent choose an optimal bundle of goods and then see what quantity of each good would be needed to satisfy each and every agent.

As in game theory the concept of equilibrium is central. One of the first definitions of equilibrium was given by Walras in 1874[3], and states that a set of prices is an equilibrium for an exchange economy if the two following properties are met

i) each agent maximizes his utility subject to his budget constraints

ii) the resulting demand equals the supply of the market.

Intuitively, at equilibrium, all the goods in the market are redistributed in the best possible way for everybody, since no agent could unilaterally gain a higher utility.

[3] L. Walras, *Elements of Pure Economics*, 1874

In 1954 Nobel laureates Kenneth Arrow and Gerard Debreu[4] proved that under general assumptions such an equilibrium does exist. It is a surprising result: for any kind of market satisfying some mild assumptions, there always exists a set of prices that will balance demand and supply of goods.

The proof given by Arrow and Debreu relies on the one given by Nash for the existence of a Nash equilibrium in strategic games. Specifically, starting from an exchange economy, Arrow and Debreu describe a strategic game, and then show that the equilibria of that game are precisely the Walrasian equilibria in the economy. The key point in transforming the economy into a game is the introduction of an extra player, whose set of strategies is the set of all possible prices. This gives some intuition on the main difference between games and markets; while in games the equilibrium is defined for each agent based on all the other agents' strategies, in markets the equilibrium is defined in terms of prices. Given a set of equilibrium prices every agent needs to know nothing more about the rest of the market: he will just solve his optimization problem, and, since the prices are equilibrium prices, is guaranteed that there will be enough goods to satisfy his requests.

Among the differences between markets and games, and the suitable equilibria, one of particular interest is tied to the concept of social optimum. A different solution concept for a game would be to consider a strategy for each player such that it is not possible for someone to be better off without some other player being strictly worse off. Such a solution, instead of considering each player individually, tries to optimize some global quantity. In a similar way, we can define an allocation of goods in a market such that each agent could not improve his situation without hurting someone else. These allocations are usually called Pareto optimal, in honor of the Italian economist who first introduced this concept.

It is a well known fact that Nash equilibria may not be Pareto optimal. The most prominent example is the Prisoner's Dilemma (see, e.g., Osborne and Rubinstein [1994]). In the Nash equilibria of this game both players receive a certain payoff, while there is an outcome in which both are strictly better off; unfortunately that solution is not itself a Nash equilibrium. On the other hand equilibria in exchange economies possess the notable property of being always Pareto optimal. This result, known as the First Fundamental Theorem of Welfare Economics (see Mas-Colell et al. [1995]), is a key point in asserting the validity of the

[4] K. Arrow, G. Debreu, *Existence of an equilibrium for a competitive economy*, Econometrica (**22**), 1954

definition of equilibrium.

Mechanism Design

Mechanism design can be thought of as the "inverse" of game theory. In game theory we are given a game and we wish to analyze it, for example by characterizing its Nash equilibria. In mechanism design we are given a set of outcomes and wish to design a game in such a way that its outcomes, for instance its Nash equilibria, satisfy the initial requests.

This corresponds to the case in which there is a central planner who can force participants to play the game but who cannot enforce the desired outcome (usually because he lacks some necessary information). In this view, the planner is interested in designing a game that, assuming the players act rationally, will lead them to the desired goal.

One typical example of mechanism design is this: assume we have to give an object to one of two players willing to pay for it. Ideally we would like to give it to the person who values it the most, but unfortunately we do not have such information. How can we design a game such that, in the solution, the object will effectively be assigned to the player valuing it the most? The game that accomplishes this is a particular kind of auction, which will be described in Section 1.1.1. Using this specific auction we are assured (if the players act rationally) that we will assign the object to the right person. In fact it is possible to prove that the best a player can do when participating in this auction is to declare his "true" valuation. So, by imposing certain rules, we have been able to assure that we will achieve the desired goals.

In the above example a different solution concept is used, rather than Nash equilibrium: the dominant strategy equilibrium. A dominant strategy for a player is an action that is always best for him, no matter what all the other players do. A set of strategies constitutes a dominant strategy equilibrium if no player has an action yielding an outcome that he prefers, given any other action by all other players. The difference with a Nash equilibrium is that in the Nash equilibrium each agent prefers the equilibrium outcome with respect to the strategies played by others at equilibrium, while in the dominant strategy equilibrium every player prefers the equilibrium outcome with respect to every possible strategy by all the other players.

Computational Issues

There are two complementary aspects when considering the interaction of computer science, game theory and economic theory. On one side there are the numerous applications of concepts from game theory and economics to new problems in computer science, on the other there are interesting algorithmic and theoretical issues in game theory and economics.

Computer Science in Economics and Game Theory The most notable algorithmic issue in both game theory and economics is tied to the core concept of equilibrium in these fields. In both cases there is a convincing definition of equilibrium, strengthened by the remarkable existence theorems due to Nash and Arrow and Debreu. Both these theorems rely on Brouwer's fixed point theorem and its extensions, which guarantee the existence of fixed points for certain kinds of functions. The problem with these theorems is that their proofs are highly non-constructive; that is they prove that a fixed point exists but do so without showing how to find one. Thus, although we are guaranteed an equilibrium exists in certain markets and games, we have no way to compute it.

The computation of equilibria is a long standing problem; Walras himself proposed a method to find equilibrium prices in an exchange economy. There have been, starting from the 60's, some interesting path-following algorithms to find equilibria in a general setting (see Todd [1976] for a survey). However, these algorithms are inherently inefficient, or converge only for a restricted class of markets.

The main contribution from computer science has been to frame the equilibrium problems in a computational complexity setting, and subsequently showing that a general and efficient algorithm for computing equilibria is unlikely to exist. The first thing to notice is that the decision version of the equilibrium problems are uninteresting, since we know an equilibrium exists, and so NP-completeness is not an appropriate concept of complexity. We have to turn to FP and FNP, complexity classes analogous to P and NP which deal with function problems: problems that require an answer more complex than just yes/no. The guaranteed existence of an equilibrium makes the problem determining one a total search problem; the class of total search problems is called TFNP, and TFNP$\subseteq$ FNP. The equilibrium problems lie in an important subclass of TFNP, a class whose definition is based on the nature of the existence proof that makes a certain problem total. This class is called PPAD[5],

[5] For Polynomial Parity Argument Directed.

and the totality of the problems in it relies on a simple combinatorial lemma: the parity argument[6]. It has been shown that the market equilibrium problem, and the Nash equilibrium problem, are both complete for the class PPAD, meaning that all other problems in PPAD can be reduced to them. The class PPAD contains many other problems for which important lower bounds are known[7], thus making the existence of efficient algorithms highly unlikely.

Economics and Game Theory in Computer Science One of the first applications of concepts and ideas from economics and game theory have been to network problems. In 1997, Kelly Kelly [1997], addressed the problem of congestion control through charging. In his setting each user in the network is charged for the bandwidth consumed, and Kelly defines an optimization problems that describe this situation. It turns out that this model corresponds to a particular market, whose equilibria are precisely the solutions of the optimization problems posed by Kelly. After his work many efficient algorithms have been developed for network and routing problems which are special cases of the market equilibrium problem.

Another fruitful application of game theory has been the "price of anarchy". This notion relates the worst-case Nash equilibrium to the global optimum (i.e. the maximum sum of payoffs achievable). It gives a measure of the inefficiency introduced in the system by allowing each agent to behave selfishly, as opposed to having a centralized control. In the context of large networks, it is often impossible to have a central authority that manages bandwidth allocation as to maximize total satisfaction. The price of anarchy is, in this case, a useful tool in analyzing the performance of different routing protocols.

Bounded Rationality

As we have seen, one of the core assumptions in game theory and microeconomic theory is that agents are fully **rational**. It can be seen how this assumption implies that the agents are able to compute all the possible consequences of their actions, taking into account the objectives and strategies of all their opponents. Intuitively this is sometimes an unrealistic assumption. For instance, from a game theoretic point of view, the games of tic-tac-toe and chess are similarly uninteresting: players need

[6] The parity argument simply states: "any finite graph has an even number of nodes with odd degree".

[7] Such as the problem of finding a Brouwer fixedpoint.

just to explore the tree of all possible moves and choose their optimal strategy. However, in real life, there is a huge difference, since there is no way for a human (nor, as of yet, for a computer algorithm) to effectively explore the tree for the game of chess for more than a couple of levels in depth.

It is fairly easy to construct examples in which the strong implications of the full rationality assumption have a computational flavor. For example consider the factoring game (see Fortnow and Santhanam [2010]). This is a two player game in which Alice communicates to Bob an integer number. Bob wins the game if he can decompose it in prime factors, otherwise Alice wins. Since every integer number can be decomposed in prime factors, Bob always wins. However, FACTORING is believed to be computationally hard, so that, in the real world, it is likely that Alice would win.

The first computational approaches to bounded rationality have been in the field of repeated games. The players' boundedness was imposed by limiting their strategies to finite automata of different sizes (see Kalai [1987]). Only more recently has this problem been considered under a more general computational complexity perspective (see Papadimitriou and Yannakakis [1994], Halpern and Pass [2010], Fortnow and Whang [1994]). However none of the models proposed has been a definitive answer to the problem, which still has many fundamental and interesting open questions.

Related Work

The literature on sponsored search auctions is huge. We give here a brief overview focusing on the parts most related to this thesis. For a general overview see, for instance, Lahaie et al. [2007] or Aggarwal and Muthukrishnan [2008].

Edelman et al. [2007] and Varian [2007] study the theoretical model behind the mechanisms in use by most search engines to sell advertisements: the generalized second price auction. Both these papers independently prove two important characteristics of the generalized second price auction:

- when there are multiple slots, the generalized second price auction is not truthful (i.e. bidding ones true value is not a dominant strategy), however
- there always is an equilibrium in which the ranking and payments are the same as in the much studied Vickrey-Clarke-Groves auction (which is known to be truthful).

Aggarwal et al. [2006] study a generalization of the model considered in Varian [2007] and Edelman et al. [2007], and show how the Vickrey-Clarke-Groves mechanism might not be truthful in this setting. They then define and characterize the unique truthful mechanism for their model. More recently Babaioff and Roughgarden [2010] showed that the generalized second price mechanism is not the only one that admits Vickrey-Clarke-Groves-like equilibria, however, among all these mechanisms, it is, in a formal sense, the simplest one.

Another line of work has been concerned with studying other kinds of equilibria in the generalized second price and their welfare guarantees as compared to the optimal mechanisms, i.e. their price of anarchy. Gomes and Sweeney [2009] study the Bayes-Nash equilibria of the generalized second price, and Leme and Tardos [2009] give the first bounds on the price of anarchy.

Most of the precise details on the workings of these auctions are kept secret by the search engines, probably in fear that they might be used by malicious users or spammers to game the system. However some experimental results carried out by search engines themselves or using their data have been published. Ostrovsky and Schwarz [2010] conduce a field experiment in which the auctions relative to a small percentage of keywords are carried out using optimal reserve prices.[8] Varian

[8] In an auction with reserve price r, the item on sale won't be allocated unless there is a bid larger than r.

[2007] complements his theoretical results by showing how they compare to some actual bids taken from search engine logs. Kitts et al. [2005] present some log data, focusing on conversion rates, price changes and click frauds.

The Tradings Agent Competition[9] is an annual event in which autonomous agents compete in different simulated markets. Since 2009 (see Jordan and Wellman [2010]) also a sponsored search market is simulated. The focus of these simulations is on the agents' strategies (see Pardoe et al. [2010], Kitts and LeBlanc [2004]). On the other hand, there has been comparatively little work on simulating a large scale sponsored search market. Previous research on agent-based simulation of adwords markets by Mizuta and Steiglitz [2000] was centered on studying the interaction of different classes of players according to their bidding time profiles, e.g. early vs late bidders. Kitts and LeBlanc [2004], along the implementation of their trading agent, describe a simulator for ad auctions used to test these agents. The main objective of the simulator is to compare several bidding strategies, e.g. random bidding vs. bid to keep relative position.

Our Results

This thesis is divided into three main parts. In a first, theoretical, one, we discuss a generalization of the classical sponsored search model. In particular we account for the fact that search engines have an almost total control over the ranking and payments of the agents. In fact, incoming bids are multiplied by "quality scores", and ranked according to this scaled order. We present an analysis of the equilibria of sponsored search markets in which the quality scores are not related to the click through rates. The generalized second price auction has similar properties in this setting; in particular it retains an efficient equilibrium. However, a fundamental difference, is that that the efficient mechanism in this setting (i.e. the Vickrey-Clarke-Groves auction) maximizes the social welfare with weights corresponding to the quality scores. Since we are assuming these are not correlated with the click through rates and they are unknown to the agents (which have no control over them), it might be the case that, from an advertiser's point of view, what is being optimized is not the actual social welfare, but the search engine's representation of it. We also consider the effect of strategic behaviors. In particular we focus on bidding rings: coalitions of bidders that act collusively. This behavior is particularly hard to uncover, especially in this

[9] http://goo.gl/PcLfg

setting, where a user might himself create multiple accounts without the seller (i.e. the search engine) never finding out. However we show that, despite the apparent danger and ease of applicability, bidding rings are not always profitable in sponsored search auction. Furthermore, their profitability is strictly connected to the actual click through rates, so that there is little agents (or the search engine) can do to make them profitable (or limit them).

In a second part we present the details and experimental results obtained with a large scale sponsored search auctions simulator that was developed as a part of this thesis. The main focus of the simulator is to analyze global effects on the market when a fraction of the agents engages in different kinds of strategic behaviors. We focus mainly on two such behaviors

- **keyword spreading** in which each agent optimizes his set of keywords, essentially looking for a set of synonyms that will yield the same number of clicks but at a smaller price,
- **starvation** in which agents increase their bids to augment their opponents' cost.

In the first case the experimental evidence points at the fact that keyword spreading is beneficial for all the participants in the market. This is due to an increased efficiency, and in fact most search engines already suggest alternative keywords to their advertisers. A by-product of our simulations in this setting is an automated and reliable procedure for finding a good set of synonyms, so that agents could employ it without using the suggestions made by the search engine.[10]

The simulations on the starvation behavior, on the other hand, show that its effects are less dramatic than it might be initially suspected. One of the deciding factors is that, for an agent, determining the highest bid that will harm his opponents without hurting his own profits is not an easy task. However these results should be taken with a grain of salt: they depend heavily on the budget constraints, after all their whole purpose is to deplete an opponent's budget as fast as possible in order to exclude him from the auction and gain his position. The role of budget constraints in this kind of environment is not so clear-cut: most search engines allow advertisers to change their budgets frequently, so seeing them as hard constraints is a very rough approximation.

[10] Some knowledge from the search engine is however assumed: in particular we assume the agents have access to somewhat reliable estimates of costs for each keyword they wish to test.

While implementing the simulator, one of the main challenges has been the lack of publicly available data regarding sponsored search markets. This has partially been overcome by utilizing on-line tools that search engines make available to their advertisers in order to optimize and estimate their campaign costs. However these tools only give information related to search volumes and costs per click. Another fundamental quantity that needed to be reliably estimated was the click through rate. To this end we implemented a set of tools and algorithms that, using only publicly available data (and in particular the search engines's results pages) extracts reasonable estimates of click through rate profiles.

Finally we present some results on bounded rationality. This work, although related, is not directly connected to any of the previous topics. It was carried out while the author was visiting prof. Lance Fortwow at Northwestern Univeristy (Evanston, U.S.A.). The first results consider the problem of recovering an agent's preferences by observing only a set of choices made by such agent. Classical microeconomic theory gives conditions under which the observations are consistent with a rational agent. However, when such conditions fail, could the observations be the output of an agent that has limited computational power (and is thus, according to classical definitions, not fully rational)? This work is part of an ongoing research project.

The other results focus on the role of randomness in bounding rationality. In fact, most research on bounded rationality assumes that agents have limited time and/or space resources. We instead consider a scenario in which agents have access to limited randomness. The role of randomness in computational complexity is central, and even in game theory the ability to randomize is fundamental. We focus on a basic zero-sum two player repeated game that has no pure Nash equilibria, i.e. matching pennies. We show that, if players are computationally unbounded, the n-stage game cannot have approximate equilibria when one of the players has only a fraction of n random coins. We then consider the case in which both players are limited to strategies that correspond to algorithms that run in time polynomial in n. We show that, in this case, ε-Nash equilibria that use only n^{δ} random coins exist if and only if one-way functions exist. We then consider a case in which one agent has a limited (i.e. logarithmic) number of random coins while to other player can only use strategies that are a fixed polynomial of n. We show how, under reasonable assumptions, by employing the Nisan-Wigderson complexity pseudorandom number generators, a ε-Nash equilibrium can be sustained. Finally we also give some results for the infinitely repeated version of the game.

Thesis Outline

This thesis is structured as follows. Part I is a general introduction to mechanism design (Chapter 1) and in particular to the auctions used in on-line advertising (Chapter 2).

Part II gives some theoretical results obtained in the context of sponsored search. In particular Chapter 3 considers a generalization of the models found in the literature while Chapter 4 gives some insights on how known strategic behaviors might affect these markets.

Part III outlines the development of a large scale simulator on ad auctions, along the methodologies used to collect publicly available data on sponsored search (Chapter 5). This simulator was successively used to experimentally evaluate how different agents' behaviors might affect the market (Chapter 6).

Part IV presents work done by the author while visiting Northwestern University. Chapter 7 introduces the problem, and first results, on dealing with a non-rationalizable set of observations, while Chapter 8 presents the results on randomness in repeated games.

In the Appendix (Part V) we give some more details on the simulator's implementation (Chapters A and B).

Part I

Background

Chapter 1

Mechanism Design

The goal of mechanism design is to achieve some predetermined social objective by aggregating preferences in a strategic setting. Assume there is a set of possible outcomes $\mathcal{A}$, and that there are n agents, each characterized by a preference relation on $\mathcal{A}$. Let Σ be the set of all possible preferences on $\mathcal{A}$, then a **social choice function** is a function $f : \Sigma \to \mathcal{A}$. For instance, $\mathcal{A}$ might be a set of electoral candidates, and f might specify that the candidate preferred by the majority of the voters be chosen.

Since we are talking about social choice, and our goal is to aggregate the preferences of different individuals in a single outcome, not all functions f are reasonable candidates. For example, a social choice function that simply picks a random outcome would not be considered "fair". Two of the most basic properties that it is reasonable to assume any social choice function should satisfy are the following:

Pareto Optimality if all agents prefer outcome x to y, then f should not choose y,

Strategy-Proof all agents must prefer the outcome of f when they declare their true preference as opposed to f's outcome when they lie.

The first property guarantees some fairness: if it is clear that y is not the best choice, then f isn't allowed to pick it. The second property ensures that the outcome will actually reflect the agents' true preferences, and not some misrepresentation of them.

One of the central results in mechanism design is a negative one. Gibbard and Satterthwaite (Gibbard [1973], Satterthwaite [1975]) showed that, if the domain of preferences is unrestricted, there are no useful social choice functions that satisfy the above properties.

Theorem 1.1 (Gibbard-Satterthwaite)**.** *Let the domain of preferences be unrestricted, $|\mathcal{A}| \geq 3$, and let f be a social choice function that satisfies Pareto optimality and strategy-proofness. Then f is dictatorial.*

In a **dictatorial** social choice function, there is an agent i such that, no matter what the preference profile is, f always outputs i's top choice. Such functions are of little, if no, use, since preferences for other agents are simply ignored. Thus, any social choice function must either be dictatorial or fail on at least one of the two fundamental properties of Pareto optimality and strategy proofness.

The process implicitly described above, however, looks rather simplified: agents announce their preferences and an action is chosen by a central authority based on f. Could it be that, by allowing the agents more complex strategies, the model would be different (and maybe negative results would not apply there)? It turns out that, in equilibrium, nothing is gained by allowing agents to adopt complex strategies. This important result is knows as the **revelation principle**. We briefly outline it here.

Assume that each agents' preferences are characterized by a **type** t_i, which defines all the private information agent i has, and let T_i be the set of allowable types for agent i. An agent also has a set of possible strategies S_i, which represent all the ways in which he might decide to interact with the central authority. A **mechanism** $\mathcal{M}$ is defined as an outcome function $g : S_1 \times S_2 \times \cdots \times S_n \rightarrow \mathcal{A}$. Once agents learn their type, they pick a strategy and communicate it to the mechanism, which will then use g to determine the corresponding outcome. In the scenario discussed in the previous paragraphs, the agents' strategy sets are limited to their type, i.e. $S_i = T_i$, and all an agent can do is report a representation of his preference. These mechanisms are called **direct**.

A mechanism $\mathcal{M}$ induces a game in which S_i is the strategy set of agent i, $\succeq_i$ his preference relation and g describes the payoffs. We say that a mechanism $\mathcal{M}$ **implements** a social choice function f if there is an equilibrium strategy profile $(s_1(t_1), \cdots, s_n(t_n))$ such that $g(s_1(t_1), \cdots, s_n(t_n)) = f(t_1, \cdots, t_n)$. A mechanism $\mathcal{M}$ **truthfully implements** a social choice function f, if there is such an equilibrium with the extra requirement that $s_i(t_i) = t_i$ for all i, i.e. each agent's strategy is simply to reveal his actual private information. Such mechanisms are also called **incentive compatible**.[1]

[1] The notion of incentive compatibility is equivalent to the notion of strategy proofness introduced earlier.

Notice that the notion of implementation crucially relies on the definition of equilibrium. There are many notions of equilibrium in game theory, but we will focus on one of the strongest ones: **dominant strategy** equilibrium.[2] We adopt the conventional game theoretic notation, in which $s_{-i} = (s_1, \cdots, s_{i-1}, s_{i+1}, \cdots s_n)$ represents the strategies of all players besides i. Now, the set of all strategies can be expressed (from i's point of view) as $s = (s_i, s_{-i})$. A strategy s_i is a dominant strategy for player i if

$$g(s_i(t_i), s_{-i}(t_{-i})) \succeq_i g(s_i'(t_i), s_{-i}(t_{-i})) \quad \forall s_i',\ s_{-i}.$$

In words, player i's utility is maximized when playing s_i, no matter what the other agents' actions are. A strategy profile is a dominant strategy equilibrium if every strategy is a dominant one for the corresponding player. We can now state the revelation principle.

Theorem 1.2 (Revelation Principle)**.** *Let f be a social choice function. If f can be implemented in dominant strategies by a mechanism $\mathcal{M}$, then there is a direct mechanism $\mathcal{M}'$ that truthfully implements f in dominant strategies.*

Proof. Let $(s_1(t_1), \cdots, s_n(t_n))$ be the dominant strategy equilibrium in $\mathcal{M}$. The direct mechanism $\mathcal{M}'$ works as follows. Let $(t_1', \cdots, t_n')$ be the types that $\mathcal{M}'$ receives as input; $\mathcal{M}'$ will simulate $\mathcal{M}$ and play $s_i(t_i')$ for agent i. Now, the fact that s_i was a dominant strategy implies that declaring their true type t_i must be a dominant strategy in $\mathcal{M}'$. Since $\mathcal{M}$ implemented f, so does $\mathcal{M}'$. For more details on the proof see Nisan [2007], Mas-Colell et al. [1995], Vohra [2011]. □

The revelation principles implies that we loose nothing in generality by restricting ourselves to direct and truthful mechanisms: the Gibbard-Satterthwaite result extends to indirect mechanisms as well.

Thus, if we wish to implement non-dictatorial social choice functions by truthful mechanisms, we must either restrict agents' preferences or look for weaker form of equilibrium in the definition of implementation. Auction theory considers the former approach, focusing on **quasi-linear** preferences.

[2] Since the definition of incentive compatible depends on the type of equilibrium used, the precise naming would be "truthfully implementable in dominant strategies" or "dominant strategy incentive compatible". However, since we will focus only on this notion of equilibrium, we will omit the "dominant strategy" portion.

1.1 Quasi-linear Preferences

Formally quasi-linear functions are functions that are linear in at least one of their arguments, i.e. functions that can be written as $f(x_1, \cdots, x_l, m) = g(x_1, \cdots, x_l) + m$. Economists usually interpret the good m as money. Mechanisms with quasi-linear preferences are often referred to as **auctions**.

In this setting, we can view the possible outcomes as a set $\mathcal{A}$, describing the possible allocation of goods that are not m, and a set of non-negative real values from $\mathbb{R}^n_+$ specifying the associated monetary transfers. Each agent is still characterized by a type $t_i \in T_i$ that defines his preferences. Without loss of generality, we will identify agents' type with their valuation (or utility) function. Thus, each agent can equivalently be characterized by a valuation function $v_i : \mathcal{A} \to \mathbb{R}$.

We can now think of the mechanism as composed of two parts

i) the allocation rule $f : V_1 \times \cdots \times V_n \to \mathcal{A}$

ii) the payment rules $(p_1, \cdots, p_n)$, where each p_i specifies the payment to player i; $p_i : V_1 \times \cdots \times V_n \to \mathbb{R}$.

Given valuations $(v_1, \cdots, v_n)$, the utility of agent i will be

$$u_i(v_1, \cdots, v_n) = v_i(f(v_1, \cdots, v_n)) - p_i(v_1, \cdots, v_n). \tag{1.1}$$

In what follows we are always going to restrict our attention to a particular class of payment rules, in which agents that don't get allocated any object or service pay nothing. The corresponding class of auctions are sometimes referred to as **normalized**.

1.1.1 Vickrey Auctions

Consider an auction of a **single item** among n agents, and assume the social function we want to implement is "give the item to the agent with the highest value".

The set of possible outcomes is $\mathcal{A} = [1..n]$, with outcome j meaning that the object was allocated to agent j. Furthermore assume that each agents' valuation can be characterized by a single value w_i (this is called a **single-parameter** setting). Agent i's valuation function is

$$v_i(a) = \begin{cases} w_i & \text{if } a = i, \text{ i.e. } i \text{ wins the auction} \\ 0 & \text{otherwise.} \end{cases}$$

Since we are considering only direct-revelation mechanisms, the agents will submit a single value, a **bid**, to the auctioneer.

Probably the most obvious auction is the first price auction, in which the winner pays exactly what he bid. It is easy to see that in this kind of auction an agent i is tempted to bid less than his valuation v_i. In fact if he wins by bidding $b_i = w_i$ he will receive a 0 payoff, while if he bids less than w_i and is still the highest bidder he will have a positive payoff. On the other hand, if having lowered his bid puts him in an inferior position, he will attain a 0 payoff; in both cases his payoff is greater or equal to 0, while before it was 0. Thus, the first price auction is not incentive compatible.

The mechanism that truthfully implements our social choice function in this setting was first discovered by W. Vickrey (in Vickrey [1961]). The auction is easily described

i) allocation rule: give the item to the highest bidder,

ii) payment rule: only the winning agent pays, and his payment is equal to the next-highest bid.

This auction is commonly known as **second-price** auction, since the winner pays the second bid. Let b_i be the bid from player i, and let b_i^* be the i-th highest bid (so that $b_1^* > b_2^* > \cdots > b_n^*$). Now bidder's i utility is

$$u_i(a) = \begin{cases} w_i - b_2^* & \text{if } a = i \text{, i.e. } b_1^* = b_i \\ 0 & \text{otherwise.} \end{cases}$$

We now give the proof that bidding ones true valuation (i.e. $b_i = w_i$) is a dominant strategy in Vickrey's auction.

Theorem 1.3. *In a second price auction, for every agent i with valuation v_i and every b_i, let u_i be i's utility if he bids v_i and u_i' his utility if he bids b_i. Then $u_i \geq u_i'$, independently of other players' actions.*

Proof. Pick any agent i, and assume he stated his true valuation, i.e. $b_i = v_i$. We can distinguish two cases:

i **won the auction** thus i got the object at the second highest declared price, $p = b_2^*$. This gives a payoff of $b_i - b_2^* \geq 0$ (since $b_i = b_1^* \geq b_2^*$). As long as i states a bid greater than b_2^* the final price, and thus his payoff, does not change. If he states a bid lower than b_2^* he will loose the auction (since the highest bid will then be b_2^*) and receive a 0 payoff.

i **did not win the auction** and so i receives a payoff of 0. As long as i bids less than b_1^* (the winning bid) his payoff will remain 0. The only way in which i could change his payoff would be by bidding $b_i' > b_1^*$. In this case he would win the object, receiving a payoff $u_i \leq v_i - b_1^*$ (since he is the winner and he bid more than b_1^* the second highest bid is now at least as large as b_1^*). As $b_1^* \geq b_i = v_i$ it follows that $u_i \leq 0$.

So, in any case, if i reports less (or more) than his true value he will receive a payoff that is surely not greater than the one he receives if the bids v_i, independently of other players' actions. □

The key fact that makes the above theorem valid is that, in a second price auction, the price paid by the winner is not determined by his own bid.

Corollary 1.4. *The second price auction truthfully implements the social choice function that assigns the object to the agent with the highest valuation.*

Proof. By Theorem 1.3 there is a dominant strategy equilibrium in which bidders bid their true value, and by the allocation rule used, in this equilibrium the bidder with the highest value gets the object. □

1.1.2 The Vickrey-Clarke-Groves Auction

The major limitation of the Vickrey auction is that it applies to single-parameter single-object settings. There is a generalization of the idea underlying Vickrey's auction to much more general scenarios, known as the Vickrey-Clarke-Groves mechanism, or VCG, accredited jointly to Vickrey [1961], Clarke [1971], and Groves [1979].

The VCG mechanism can be applied when there are m heterogeneous goods for sale and agents have arbitrarily complex preferences, represented by valuation functions $v_i(\cdot)$. The social choice function being implemented is the one maximizing social welfare. Thus each agent reports a valuation function $b_i(\cdot)$ to the auctioneer, which chooses an assignment $a \in \mathcal{A}$ such that

$$a \in \arg\max_{a' \in \mathcal{A}} \sum_{i=1}^{n} b_i(a').$$

Let $\mathsf{OPT}(b) = \mathsf{OPT}(b_1, \cdots, b_n)$ denote the value of this assignment (i.e. the optimal social welfare) and, for notational convenience, let

- $\mathsf{OPT}_{-i}(b) = \mathsf{OPT}(b) - b_i(a)$,

- $\mathsf{OPT}(b_{-i}) = \mathsf{OPT}(b_1, \cdots, b_{i-1}, b_{i+1}, \cdots, b_n)$.

Intuitively OPT_{-i} represents the value of the optimal surplus without i's contribution, while $\mathsf{OPT}(b_{-i})$ is the maximum social welfare that *could* have been obtained had i not participated in the auction. Notice that, most importantly, both terms do *not* depend on i's valuation.

The crux of the mechanism lies in the payments, which are defined as

$$p_i = \mathsf{OPT}(b_{-i}) - \mathsf{OPT}_{-i}(b).$$

In essence each agent is paying an amount equivalent to all the other bidders' losses *due to his presence*. Economists call this **negative externalities**. One of the main properties of the VCG mechanism is that it is incentive compatible.

Theorem 1.5. *Truthful reporting is a dominant strategy in VCG.*

Proof. Pick any bidder i, and any fixed profile of valuations for the other bidders $\{b_j\}_{j \neq i}$. Let a be the allocation when bidder i reports truthfully and $\overline{a}$ the allocation when i reports a valuation function that is not v_i. Notice that in both cases, the payment for i is the same (since the other agents' reports are fixed). When i does not report truthfully, his utility is

$$\begin{aligned} u_i(\overline{a}) = v_i(\overline{a}) - p_i &= v_i(\overline{a}) - \mathsf{OPT}(b_{-i}) + \mathsf{OPT}_{-i}(b) \\ &\leq v_i(a) + \mathsf{OPT}_{-i}(b) - \mathsf{OPT}(b_{-i}) \\ &= v_i(a) - p_i = u_i(a), \end{aligned}$$

where the inequality follows since, by definition, $v_i(a) + \mathsf{OPT}_{-i}(b) = \mathsf{OPT}(b)$ is the value of the allocation that maximizes social welfare. Thus, in particular, it must be the case that $\mathsf{OPT}(b) \geq v_i(\overline{a}) + \mathsf{OPT}_{-i}(b)$. □

The VCG mechanism (and the proof of Theorem 1.5) do not impose any restriction on the format of bidder's preferences (of course, they have to be quasi-linear). Thus it is always possible to truthfully implement the social choice function that maximizes social welfare.

One might ask which other social choice functions can be implemented. We first give an answer to this question in the single-parameter setting, and then extend it to the more general cases.

Incentive compatible Characterization in Single Parameter Settings

It turns out that, in single parameter domains, it is possible to fully characterize incentive compatible mechanisms. The class of social choice functions that can be implemented is fairly large: the only condition is a monotonicity one. Recall that, in single parameter settings, $\mathcal{A} = [1..n]$. A social choice function f is called **monotone** if, for all i, having fixed v_{-i}, if $f(v_i, v_{-i}) = i$ then it must also be the case that $f(v_i', v_{-i}) = i$ for all $v_i' \geq v_i$. In words: if the valuation of a winning agent increases, all other values being fixed, he must still win the auction. Since the agents only value "winning", the social choice function from i's viewpoint is a step function (see Figure 1.1). We call c_i bidder's i **critical value**, and identify it with the minimum value i can have and still win the object. Formally $c_i(v_{-i}) = \sup_{v_i | f(v_i, v_{-i}) \neq i} v_i$. We first show that, in single parameter incentive compatible mechanisms, all winning agents must pay the same value.

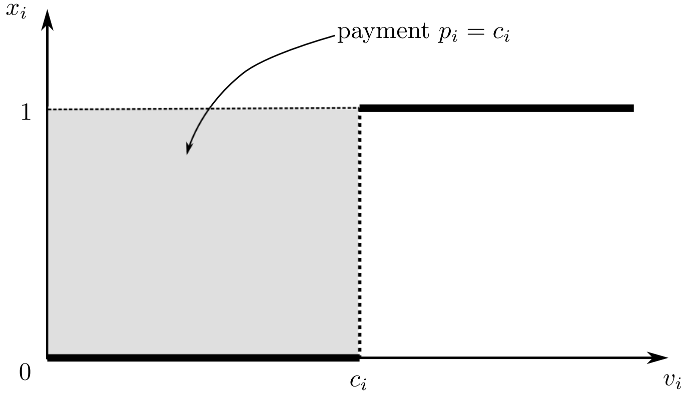


Figure 1.1: A monotone social choice function in a single parameter setting, for fixed v_{-i}.

Lemma 1.6. *In an incentive compatible mechanism $\mathcal{M} = (f, p_1, \cdots, p_n)$ in a single parameter setting, all winning bids pay the same amount p.*

Proof. Assume not. Then there is a winning bidder that would prefer changing his bid to that of another winning bidder that is paying less. This contradicts the incentive compatible assumption. □

We now give the full characterization of incentive compatible mechanisms in single parameter domains.

Theorem 1.7. *A normalized mechanism $\mathcal{M} = (f, p_1, \cdots, p_n)$ in a single parameter setting is incentive compatible if and only if* *i) f is monotone*

ii) for all i, $p_i = c_i$ (i.e. each bidder pays his critical value).

Proof. (If part) This part is similar to the proof of Theorem 1.3 [page 20] (notice that in a single item auction, the second highest bid is the critical value). For any bidder i, and fixed v, his utility is $v_i - c_i(v_{-i})$ if he wins, and 0 if he looses. By monotonicity of f, and assuming i is winning, he is better off bidding v_i, since if he bids less he might loose the auction and decrease his utility (if he bids more his utility does not change). On the other hand, if i is loosing, he is still better off announcing v_i, since if he announced more he might win (decreasing his utility), while if he bids less nothing changes (again, this holds because since f is monotone).

(Only if part) First we show that, if $\mathcal{M}$ is incentive compatible, then f is monotone. Assume not. Then there must be an i and $v_i' > v_i$ such that $f(v_i, v_{-i}) = i$ and $f(v_i', v_{-i}) \neq i$ (i.e. by increasing his declared value i ceased being served). Since we assumed the mechanism was incentive compatible, we know that i is better off bidding v_i than v_i', so that he prefers winning to loosing, i.e. $v_i - p_i(v_{-i}) \geq 0$. Now assume i's true value was v_i'; then he would prefer loosing to winning, i.e. $v_i' - p_i(v_{-i}) \leq 0$ (notice that since we changed only i, his payments are the same). Thus it must be the case that $v_i = v_i'$, a contradiction.

Now we show that agents must pay their critical values. Assume, by way of contradiction, that a winning agent i is paying $p > c_i(v_{-i})$. By Lemma 1.6 we know that all winning agents are paying p. So, an agent with value v_i' such that $c_i(v_{-i}) < v_i' < p$ would be better off declaring a lower value, which contradicts the incentive compatible assumption. The other direction (i.e. when $p < c_i(v_{-i})$) is similar. □

Incentive compatible Characterization in General

When we impose no restriction on the domain of preferences, it can be shown that, in a sense, very little *besides* the VCG is incentive compatible. We just state these theorems here, referring to Vohra [2011], Nisan [2007] for more details.

A social choice function f satisfies **(weak) monotonicity** if, for all i and v_{-i} we have that $f(v_i, v_{-i}) = a \neq b = f(v_i', v_{-i})$ implies that $v_i(a) - v_i(b) \geq v_i'(a) - v_i'(b)$. In words: if the allocation changes solely because i changed his valuation, it must be the case that i changed the relative value of the new choice.

Theorem 1.8. *If a mechanism $\mathcal{M} = (f, p_1, \cdots, p_n)$ is incentive compatible, then f satisfies weak monotonicity. If all domains of preferences V_i are convex* *sets, then for every social choice function f that satisfies weak monotonicity, there exist payment rules $(p_1, \cdots, p_n)$ that makes the mechanism incentive compatible.*

There is also another characterization given by Roberts (in Roberts [1979]). In Roberts' theorem it is easy to verify that the only incentive compatible mechanisms in general settings are weighted versions of VCG. The **weighted VCG** mechanism assigns weights w_i to each agent, and the objective is to maximize the weighted social welfare. It is not difficult to see that the following payment rule (along choosing the optimal allocation) makes the mechanism incentive compatible:

$$p_i = \frac{1}{w_i}\left(\mathsf{OPT}(v'_{-i}) - \mathsf{OPT}_{-i}(v')\right), \tag{1.2}$$

where, if a is the optimal weighted allocation, $\mathsf{OPT}(v') = \sum_{j \in a} w_j v'_j(a)$. Such functions are called **affine maximizers**. More formally, f is an affine maximizer if, for some player weights w_i and outcome weights c_a, we have that $f(v_1, \cdots, v_n) = \arg\max_{a \in A'}(c_a + \sum_j w_j v_j(a))$. Roberts' theorem shows that being an affine maximizer is a necessary condition for being incentive compatible.

Theorem 1.9 (Roberts' Theorem). *If $|\mathcal{A}| \geq 3$, f is onto, $V_i = \mathbb{R}^{\mathcal{A}}$ for all i and f is incentive compatible, then f is an affine maximizer.*

1.1.3 Some drawbacks of VCG

As the characterization in the previous section shows, VCG plays a central role in theoretical mechanism design, however it is not often used in practice. There are many reasons for this, Ausubel and Milgrom [2006] list several of these. We will focus on just two of them that have a computational flavor, and for ease of exposition we will restrict attention to the setting of **combinatorial auctions**. In this scenario, a set of goods m is on auction, and there are n interested agents.

First assume bidders are **single-minded**, i.e. they have interest for one particular subset of items: bidder i has value v_i for receiving items $S_i \subseteq [1..m]$ and value 0 otherwise. There is just one copy of each item, so we can't allocate their desired sets to bidders i and j unless $S_i \cap S_j = \emptyset$. To simplify even further assume that all players receive the same utility from getting their allocation (i.e. $v_i = v_j = 1 \forall\ i, j$). If we needed to find an incentive compatible mechanism that maximizes social welfare for this setting the answer would be easy: just use VCG. Furthermore, since this is a single parameter setting, we just need to find the allocation that maximizes $\sum_i v_i$ and the payments are simply the critical values. However, the problem of computing the optimal allocation can be rewritten as follows:

> given a collection $S_1, \cdots, S_n$ of finite subsets of $[1..m]$, find the maximum number of mutually disjoint sets.

This is the optimization version of the set packing problem, which is known to be NP-complete (see Garey and Johnson [1979]). Thus, although we have a mechanism that is guaranteed to implement our social choice functions, and also a formula for the payments, it might be computationally infeasible to determine who gets which objects.

Another computational problem arises if we slightly generalize the above setting and allow agents to have more complicated preferences. Assume for instance that each agent has a value for every subset of $[1..m]$. The VCG mechanism can still be used, however its implementation requires that agents report their complete preferences to the central authority. This involves, in the worst case, an exponential amount of communication.

Chapter 2

On-line Advertising

2.1 Auctions in On-line Advertising

The history of advertising on the Web starts with a model related to the physical world: an advertiser pays a certain website to display his ad. To adapt the payment to the size of the audience, the advertiser is charged a small amount each time the ad is viewed (pay-per-impression). This, as in the physical world, is detrimental for small advertisers in niche markets, who are in competition for ad space with big companies. In the real world a niche advertiser could partially overcome such a problem by placing ads only in certain locations, e.g. in specialized journals. This same idea can be ported to the Web, but in a much more general setting. Without negotiating deals with every website or portal related to his product, there is a one stop solution for every advertiser: the search engine.

When a user enters a web query, from an advertiser's point of view, he is expressing interest for that particular product or service. So the results page seems an obvious place to display ads regarding that particular keyword: this is the basic concept behind keyword based advertising. This idea was so successful, for both search engines and advertisers, that it is today a 15$ billion per year business, and it is the primary source of income for most search engine companies.[1]

Another turning point in the online advertising business has been the switch in payment modes; from pay-per-impression to **pay-per-click**. In the Web, the intentions of a user, a potential customer, can be further

[1] See `https://secure.wikimedia.org/wikipedia/en/wiki/Search_engine_marketing`.

verified; we know someone is interested in our product when his web search contains certain keywords, but we are virtually sure he is interested when he actually clicks on our ad. Sponsored search today relies on this model, and an advertiser is charged only when someone clicks on his ad (pay-per-click). At first sight this could seem a less profitable design for the search engine; after all an ad is very often displayed without being clicked. It turns out that, being able to pay to advertise only to very plausible customers has a huge value to companies, driving the price of each click to high values (tens of dollars in certain cases). This model was so successful that today it is adopted by all the major search engine companies.

If in the pay-per-impression model the prices were fixed, in the pay-per-click model they are determined dynamically. This happens because it would be unreasonable for the search engine to come up with a set of prices for each possible search query, and furthermore the prices should reflect the actual competition for a given keyword at any time. Furthermore the entities being sold, i.e. clicks, have an unclear nature, and it is not obvious how they should be priced. The determination of a price for a good of unknown (or unclear) value is a well known problem in economics, and can often be solved by a simple mechanism: an auction. This is the solution adopted by most search engines today, and the resulting systems have been called **ad auctions**.[2]

When the advertisers subscribe to the service they specify a number of keywords in which they are interested, and for each of these they declare a how much they are willing to pay for each click. Advertisers are also requested to specify a maximum daily budget. If there were a single slot in which to display ads, this setting would resemble a single-item auction; upon the arrival of a certain query, the search engine selects the interested bidders, and displays the ad of the one with the highest bid. This advertising model is generally referred to as **keyword based advertising**.

2.2 A Model for Position Auctions

We consider a fixed search engine query, i.e. a single auction. Let there be k available slots and n advertisers. We will usually assume that $n > k$: this can be done without loss of generality, since if there were $n \leq k$ bidders we could define an equivalent auction with $n' = k + 1 - n$ ghost

[2] These auctions go under a number of different names. Economists often refer to them as **position auctions**, while in computer science the term **sponsored search auctions** is often used.

bidders whose valuation and bids are all 0. Each agent is assigned to a slot and, assuming agent i is in slot j, each agent/slot combination is characterized by its **click through rate**, which we denote as $\mathsf{CTR}_{i,j}$. This value can be interpreted as the expected number of clicks the ad receives in a fixed amount of time (an hour, a day, a month). These values can be estimated by the search engine from historical data. A usual assumption is that $\mathsf{CTR}_{i,j}> \mathsf{CTR}_{i,j+1}$ for any i, j. This means that, for any agent, higher slots give more clicks. An immediate implication of this assumption is that all agents' interests are aligned: everybody has the same preference ordering on slots.

Consider a simplified version of the online advertising problem in which there is just one slot available for displaying ads. Since we are considering a single keyword, this situation resembles the single item sealed bid auction described in Section 1.1.1 [page 19]. In fact, although there are multiple clicks for sale, only one bidder can get allocated to the only slot: he then will receive all the clicks. Thus agents can be characterized by a single parameter, namely their value v_i for receiving a click. Each bidder submits a bid b_i as a representation of his private information to the search engine.

In our simplified scenario once an ad is displayed it will receive a number of clicks, according to the known value of the click through rate α. The search engine sorts the bidders according to their bid value and displays the ad of the first bidder. Then, by some mechanism, a price p for each click is established. The winning agent will pay for each click, and so the revenue R at price p for the search engine is

$$R(p) = \alpha p.$$

On the other hand each click is valued v_i by the winning bidder i, so the bidders' utilities at the current price are now

$$u_i(p) = \begin{cases} 0 & \text{if } i \text{ is not displayed} \\ \alpha(v_i - p) & \text{if } i \text{ is in the only slot.} \end{cases}$$

A Note on Weights and click through rates In the most general model, each agent i is assigned a weight w_i, and agents are sorted in decreasing order of the product $w_i b_i$. The weights w_i allow the search engine to control the ranking. In the literature two possible settings for these weights have been considered

i) $w_i = 1$ for all i, so that agents are ranked by bid,

ii) $w_i = \mathsf{CTR}_{1,i}$.

in the second alternative, also known as rank-by-revenue, the search engine assigns higher weights to agents that, intuitively, generate more clicks. The fact that most of the research has focused on these two settings is mainly due to historical reasons. In fact, one of the first companies to implement sponsored search, Overture,[3] simply ranked advertisers by bid, i.e. setting $w_i = 1$ for all i. Subsequently, when Google entered the market, their auctions were run with the revenue maximizing weights, $w_i = \mathsf{CTR}_{1,i}$.

Before studying the auctions in use by search engine a further clarification must be made on click through rates. A common assumption is that click through rates can be decomposed into a slot specific and advertiser specific components, i.e. there are non-negative numbers $\alpha_1, \cdots, \alpha_k$ and $\beta_1, \cdots, \beta_n$ such that $\mathsf{CTR}_{i,j} = \alpha_i\beta_j$. If this is possible, the click through rates are called **separable**. Furthermore, if all $\beta_i = 1$, then the click through rate of a slot does not depend on the actual ad currently displayed in that slot. In this case we call click through rates **bidder independent**, and label them simply $\alpha_1, \cdots, \alpha_k$.

Notice that, when click through rates are separable, we assume that the search engine ranks bidders according to the product βb, so as to generate the highest number of clicks. In this setting, the case case in which $w_i = \mathsf{CTR}_{1,i}$ is equivalent to the case in which $w_i = 1$. In fact, in the first case the agents are sorted according to $w_i b_i$, i.e. i is above j if and only if

$$w_i b_i \geq w_j b_j.$$

Rewriting w_i as $\mathsf{CTR}_{1,i} = \alpha_1 \beta_i$ we obtain

$$\alpha_1 \beta_i b_i \geq \alpha_1 \beta_j b_j,$$

so that we can equivalently sort by $\beta_i b_i$, ignoring weights.

In what follows we will outline the model as originally studied by Edelman et al. [2007] and Varian [2007], and let $w_i = 1$ for all i. A generalization of this model will be considered in Chapter 3 [page 57].

[3] Formerly GoTo.com, Overture was acquired by Yahoo! in 2003.

2.3 Generalized Second Price Auctions

We now consider the general case, in which there are k available slots, with bidder independent click through rates. For convenience we assume that there is an extra slot $k+1$ whose click through rate is 0, so that $\alpha_1 \geq \cdots \geq \alpha_k \geq \alpha_{k+1} = 0$. We will denote the valuations, bids, prices and agents sorted by decreasing bid order as v^*, b^*, p^* and a^* respectively. So, for example, if bidder i receives the third slot, $i = a_3^*$, his valuation is $v_i = v_3^*$, his bid $b_i = b_3^*$ and the price he will pay for every click p_3^*.

Vickrey's second price auctions can be generalized in different ways to deal with multiple objects. The most natural way will be described in Section 2.4 [page 39], while now we introduce a different generalization, widely adopted in the context of online advertising; the generalized second price auction, or GSP. In this mechanism each agent that is awarded a slot will pay a price corresponding to the next highest bid. Since we assume that $n > k$, the bidder in the last slot will pay the bid of the first excluded bidder. Notice that, in the case in which there are more slots than agents and we add in the model fictitious bidders with 0 bids, the last displayed ad will pay a 0 price.[4]

Example 2.3.1. Assume there are 2 advertising slots (1 and 2) and 3 interested agents (a, b and c). The slots have a click through rate of 200 and 100 clicks per hour, respectively ($\alpha_1 = 100$, $\alpha_2 = 200$). The tree advertiser value each click $v_a = 5\$$, $v_b = 4\$$ and $v_c = 2\$$ respectively.

Slots		Agents	
Position	CTR	Agent	Value per click
1	200	a	$v_a = 5\$$
2	100	b	$v_b = 4\$$
-	0	c	$v_c = 2\$$

Assume they all bid their true valuation v_i, then a and b will have their ads displayed, paying 800$ and 200$ per hour, respectively.

Position	Agent	Bid	Price per click	Revenue
1	a	$b_1^* = b_a = 5\$$	$p_1^* = b_2^* = 4\$$	$5\$ \cdot 200 - 4\$ \cdot 200 = 200\$$
2	b	$b_2^* = b_b = 4\$$	$p_2^* = b_3^* = 2\$$	$4\$ \cdot 100 - 2\$ \cdot 100 = 200\$$
-	c	$b_3^* = b_c = 2\$$	$p_3^* = 0\$$	0$

[4] In reality search engines will charge a positive amount in this case.

Since a values 5\$ each click received, he will earn $200 \cdot 5\$ - 800\$ = 200\$$. The bidder in the second slot, b, will earn $(4 - 2)100\$ = 200\$$. ■

The GSP and the second price auction are tightly connected: in fact they coincide if there is just one available slot. This similarity could suggest that also the incentive compatibility of the second price auction (see Corollary 1.3 [page 20]) is carried over to the GSP. Unfortunately this is not the case, as shown by a slight modification of the above example.

Example 2.3.2. Player a, having bid 5\$ will win the top slot, and, as shown above, will earn 200\$ (receiving 200 clicks that he values 5\$ each and paying 4\$ for each of them). What would happen if player a decided not to bid his true valuation v_a? Suppose he bids $b_a = 3\$ < v_a$. This implies he will not be in the top slot anymore, receiving half the number of clicks. Nonetheless he will earn 500\$ (receiving 100 clicks which he still values 5\$ each).

Position	Agent	Bid	Price per click	Revenue
1	b	$b_1^* = b_b = 4\$$	$p_1^* = b_2 = 3\$$	$4\$ \cdot 200 - 3\$ \cdot 200 = 200\$$
2	a	$b_2^* = b_a = 3\$$	$p_2^* = b_3 = 2\$$	$5\$ \cdot 100 - 2\$ \cdot 100 = 300\$$
-	c	$b_c^* = v_c = 2\$$	$p_3^* = 0\$$	0\$

The payments will instead be 200\$, since the next highest bid is 2\$. Thus player a's revenue is 300\$, showing that he is better off not reporting his true valuation. ■

2.3.1 Equilibria

Although the GSP auctions do not have the strong properties of Vickrey's original mechanism, they still do have some attractive characteristics. One of them is that they can be thoroughly analyzed. To this end we will define a game based on the GSP mechanism. We will assume it is a game with complete information, that is we assume that each agent knows the valuations of all other bidders. This assumption is needed to make the analysis feasible, but it is actually not as unrealistic as it first sounds: in fact the ad auctions are repeated numerous times, with mostly the same agents, so we can suppose that, after some amount of time, each bidder has a (more or less) correct view of the others' valuations.

We are then interested in describing the equilibria of such games. For a game to be in Nash equilibrium, each player should prefer his current strategy, given the strategies of all other players. In the GSP scenario, if we fix the strategies for all other players, we are actually fixing the bids for each slot. So the equilibrium conditions should capture the fact that no agent is willing to modify his bid in order to obtain a different slot.

Definition 2.3. *A Nash Equilibrium set of prices (NE) satisfies*

$$(v_j^* - p_j^*)\alpha_j \geq (v_j^* - p_k^*)\alpha_k \quad \textit{for} \quad k > j \tag{2.1}$$

$$(v_j^* - p_j^*)\alpha_j \geq (v_j^* - p_{k-1}^*)\alpha_k \quad \textit{for} \quad k < j \tag{2.2}$$

for each slot $j = 1, \ldots, m$, where p_j^* is determined by the GSP mechanism, i.e. $p_j^* = b_{j+1}^*$.

We need two different conditions in view of the way the GSP mechanism works. Consider the examples above; if bidder a wants to move downwards he has to bid lower than the price the agent below him (b) is currently paying. While if agent b wants to move upwards he has to beat the bid of the agent above him (a). So the first condition is related to slots lower than the agent's current one; it ensures that, even at the current price of these slots he would not prefer them. The second condition can be rewritten as

$$(v_j^* - p_j^*)\alpha_j \geq (v_j^* - b_k^*)\alpha_t \quad \text{for} \quad k < j,$$

since by the GSP mechanism $p_{k-1}^* = b_k^*$. It ensures that, if i were to overbid that agent (thus paying b_k) he would not prefer the payoff in that position. Another way to view this asymmetry is this: if an agent wants to move downward he can do so without altering the current prices (he will simply swap positions with the agent below him and end up paying what that agent was paying). On the other hand, if an agent wants to move upward, the prices of the slots above him will change: in fact, he will have to overbid the agent above him, and that is precisely the bid he will end up paying (since the overbid agent will become the next highest bid).

There is an interesting subset of (NE) that has some notable properties: the equilibrium bids and prices can be recursively bounded.

Definition 2.4. *A symmetric Nash equilibrium set of prices (SNE) satisfies*

$$(v_j^* - p_j^*)\alpha_j \geq (v_j^* - p_k^*)\alpha_k \quad \textit{for all} \quad k$$

for all positions j, where $p_k^* = b_{k+1}^*$.

This is precisely the condition that holds in a (NE) for prices below the current position. Clearly any (SNE) is a (NE), since $p^*_{k-1} = b^*_k \geq b^*_{k+1} = p^*_k$. In a (SNE) each bidder prefers his current position to any other position at current prices; in particular he prefers his position to the positions above him even if he could obtain them without overbidding the current occupier (i.e. at the same price). This property is also often called **envy-freeness**, since no agent would choose to change places. Following Varian [2007] we enumerate some properties of (SNE).

Fact 2.5. *In a (SNE) each bidder pays no more than what he values each click, that is*

$$v^*_j \geq p^*_j.$$

Proof. By definition the click through rate of any slot s such that $s > m$ is[5] 0. Since the inequality characterizing the (SNE) holds for all j we can write

$$(v^*_j - p^*_j)\alpha_j \geq (v^*_j - p^*_{m+1})\alpha_{m+1} = 0,$$

hence $v^*_j \geq p^*_j$. □

Fact 2.6. *In a (SNE) bidders are sorted in decreasing valuation order,*

$$v^*_{j-1} \geq v^*_j \quad \textit{for all} \quad i.$$

Proof. The (SNE) condition can be rewritten as

$$v^*_j(\alpha_j - \alpha_k) \geq p^*_j\alpha_j - p^*_k\alpha_k.$$

Consider the (SNE) inequalities for two different positions k and j:

$$v^*_j(\alpha_j - \alpha_k) \geq p^*_j\alpha_j - p^*_k\alpha_k \tag{2.3}$$
$$v^*_k(\alpha_k - \alpha_j) \geq p^*_k\alpha_k - p^*_j\alpha_j. \tag{2.4}$$

Adding up the two inequalities (2.3) and (2.4), we obtain

$$(\alpha_j - \alpha_k)(v^*_j - v^*_k) \geq 0$$

showing that the values of α and v^* should be ordered in the same way (and we assume that click through rates are in decreasing order). □

Fact 2.7. *In a (SNE), $p^*_{j-1}\alpha_{j-1} > p^*_j\alpha_j$ and $p^*_{j-1} \geq p^*_j$ for all j. If $v^*_j > p^*_j$ then $p^*_{j-1} > p^*_j$.*

[5] Recall that there are only m slots, so $\alpha_s = 0$ for $s > m$ simply states that there will be no clicks on slots that do not exist.

Proof. The definition of (SNE) implies that

$$(v_j^* - p_j^*)\alpha_j \geq (v_j^* - p_{j-1}^*)\alpha_{j-1}$$

which can be rewritten as

$$p_{j-1}^*\alpha_{j-1} \geq p_j^*\alpha_j + v_j^*(\alpha_{j-1} - \alpha_j),$$

and assuming $\alpha_{j-1} > \alpha_j$ this implies

$$p_{j-1}^*\alpha_{j-1} > p_j^*\alpha_j.$$

To prove that $p_{j-1}^* \geq p_j^*$ consider

$$p_{j-1}^*\alpha_{j-1} \geq p_j^*\alpha_j + v_j^*(\alpha_{j-1} - \alpha_j) \geq p_j^*\alpha_j + p_j^*(\alpha_{j-1} - \alpha_j) = p_j^*\alpha_{j-1},$$

in which we used $v_j^* \geq p_j^*$ (Fact 2.5 [page 34]). Dividing the above by α_{j-1} we obtain $p_{j-1}^* \geq p_j^*$. If the second inequality is strict ($v_j^* > p_j^*$) then $p_{j-1}^* > p_j^*$. □

The next fact is probably the most interesting one. It asserts that it is only necessary to verify the (SNE) inequalities for one position above or below the one considered to show that the inequalities hold for all the slots. This condition is also referred to as **local envy freeness**.

Fact 2.8. *If a set of bids satisfies the (SNE) conditions for $i+1$ and $i-1$ then it satisfies the (SNE) conditions for all i.*

Proof. Assume there are 3 available slots, and the (SNE) inequality holds between slots 1 and 2 and between slots 2 and 3. Should it hold for slots 1 and 3 we would have given an example (easily extensible to the general case). Let's write the inequalities we know are holding:

$$v_1^*(\alpha_1 - \alpha_2) \geq p_1^*\alpha_1 - p_2^*\alpha_2 \tag{2.5}$$
$$v_2^*(\alpha_2 - \alpha_3) \geq p_2^*\alpha_2 - p_3^*\alpha_3 \tag{2.6}$$

By Fact 2.6 [page 34] we know that $v_1^* \geq v_2^*$, hence (2.6) becomes

$$v_1^*(\alpha_2 - \alpha_3) \geq p_2^*\alpha_2 - p_3^*\alpha_3$$

Adding this with (2.5) we obtain

$$v_1^*(\alpha_1 - \alpha_3) \geq p_1^*\alpha_1 - p_3^*\alpha_3$$

which is precisely the (SNE) condition for slots 1 and 3. □

Given all of the above facts we can characterize the prices at equilibrium. We know that bidder in slot j will not want to move down one slot, while bidder in slot $j+1$ will not want to move up, so

$$(v_j^* - p_j^*)\alpha_j \geq (v_j^* - p_{j+1}^*)\alpha_{j+1}$$
$$(v_{j+1}^* - p_{j+1}^*)\alpha_{j+1} \geq (v_{j+1}^* - p_j^*)\alpha_j.$$

Combining these two inequalities we obtain

$$v_j^*(\alpha_j - \alpha_{j+1}) + p_{j+1}^*\alpha_{j+1} \geq p_j^*\alpha_j \geq v_{j+1}^*(\alpha_j - \alpha_{j+1}) + p_{j+1}^*\alpha_{j+1}.$$

Substituting bids for prices according to the GSP mechanism we obtain

$$v_{j-1}^*(\alpha_{j-1} - \alpha_j) + b_{j+1}^*\alpha_j \geq b_j^*\alpha_{j-1} \geq v_j^*(\alpha_{j-1} - \alpha_j) + b_{j+1}^*\alpha_j. \qquad (2.7)$$

This last inequality bounds the bids in any (SNE), with respect to the bids of agents directly above and below him. Considering just the lower bound on the bids we can write

$$b_j^*\alpha_{j-1} = v_j^*(\alpha_{j-1} - \alpha_j) + b_{j+1}^*\alpha_j.$$

Since there are only k slots and we assume $\alpha_{k+1} = 0$ we can solve the recursion, obtaining

$$b_j^*\alpha_{j-1} = \sum_{k \geq j} v_k^*(\alpha_{k-1} - \alpha_k). \qquad (2.8)$$

We have now a complete characterization of bids (and thus prices) that yield a symmetric Nash equilibrium.

Example 2.3.9. Consider the situation in Example 2.1 [page 31], in which agents bid their true valuations. In this case these bids do not lead to (SNE) prices, since bidder a would prefer the second slot (at current prices), as this would give him a payoff of 300$: 100$ more than he is making in the top slot. This fact was exploited in Example 2.2 [page 32], where it was shown that, by changing his bid, a could actually increase his profit.

We could now ask when does truthful bidding lead to a (SNE) in the GSP mechanism. The bounds derived above can give some insight. If bidders are truthful, the upper bound of Equation (2.7) becomes

$$v_j^*\alpha_{j-1} \leq v_{j-1}^*(\alpha_{j-1} - \alpha_j) + v_{j+1}^*\alpha_j.$$

Rearranging terms we get

$$\alpha_j(v^*_{j+1} - v^*_{j-1}) + \alpha_{j-1}(v^*_{j-1} - v^*_j) \geq 0.$$

If we consider the case of Example 2.1 [page 31] for $j = 2$ we obtain

$$100(2\$ - 5\$) + 200(5\$ - 4\$) < 0,$$

which shows that in this case bidding their true valuation does not lead agents to a (SNE). But Equation (2.7) also gives lower bounds, that when bidders are truthful become

$$v^*_j \alpha_{j-1} \geq v^*_j(\alpha_{j-1} - \alpha_j) + v^*_{j+1}\alpha_j.$$

As above, rearranging terms, we get

$$\alpha_j(v^*_{j+1} - v^*_j) \leq 0.$$

By definition $\alpha_j \geq 0$, and by Fact 2.6 [page 34] $(v^*_{j+1} - v^*_j) \leq 0$ in a (SNE). So the lower bounds at (SNE) are always respected by truthful bids. This implies that, if truthful bidding is not a (SNE), then any (SNE) will have bids strictly lower than the players' valuations. ■

2.3.2 Generalizations

When we consider separable click through rates (as opposed to bidder independent click through rates as in the previous paragraphs), the model and equilibrium properties remain unchanged.

Consider first the case in which merchants are ranked by bid, so that payment for slot j is simply b_{j+1}. The equilibrium conditions (2.1) for player i currently in slot i:

$$\alpha_i\beta_i(v_i - b_{i+1}) \geq \alpha_j\beta_i(v_i - b_{j+1}). \tag{2.9}$$

which is exactly (2.1).

A similar argument holds when bidders are ranked by revenue, i.e. by decreasing order of $\beta_i b_i$. In this case the auction can't simply charge bidder i in slot i the next highest bid b_{i+1}, since it could very well be that $b_{i+1} > b_i$. In this setting payments are usually defined as "the minimum bid necessary for i to retain his position", so that

$$p_i = \frac{\beta_{i+1}b_{i+1}}{\beta_i}.$$

By using these payments, bidder's *i* utility when in position *i*

$$\begin{aligned} u_i &= \alpha_i \beta_i \left(v_i - \frac{\beta_{i+1} b_{i+1}}{\beta_i} \right) \\ &= \alpha_i (\beta_i v_i - \beta_{i+1} b_{i+1}). \end{aligned}$$

Thus, this setting is equivalent to a model with bidder independent click through rates (as discussed in previous paragraphs) in which we scale all bids and valuations by β.

2.4 Vickrey-Clarke-Groves Auctions

Let's consider again the simple case where there is only one object for sale. Of course this is just a special case of the multi object auction, but we know (see Theorem 1.3 [page 20]) that the Vickrey second price mechanism will enforce every player to bid truthfully.

Example 2.4.10. Consider the situation in Example 2.1 [page 31], but assume there is just 1 advertising slot available.

Slots		Agents	
Position	CTR	Agent	Value per click
1	200	a	$v_a = 5\$$
-	0	b	$v_b = 4\$$
		c	$v_c = 2\$$

Assume all players bid their true valuation v_i; then, applying the Vickrey second price auction, a will have its ad displayed, paying b's bid. That is a will pay $200 \cdot 4\$$ while earning $200 \cdot 5\$$, which is a payoff of 200$. Now consider the same situation, only without buyer a. In this case b would win the auction, paying c's bid, receiving 800$ worth of clicks. So, since b earns 0 if a is there, we can say that merely a's presence harms b precisely 800$. But that is exactly what a pays. As Theorem 2.22 [page 51] states a is charged according to his negative externalities. ■

The GSP mechanism was derived from the second price auction, but it did not retain its most useful property. The Vickrey Clarke Gables (VCG) mechanism (introduced in Section 1.1.2 [page 21]) generalizes the Vickrey second price auction in a different way, as illustrated by the Example 2.11. Each bidder is charged according to the negative effect he has on other participants.

Example 2.4.11. Consider once again the scenario in example 2.1 [page 31].

Slots		Agents	
Position	CTR	Agent	Value per click
1	200	a	$v_a = 5\$$
2	100	b	$v_b = 4\$$
-	0	c	$v_c = 2\$$

And assume all bidders bid their true valuation (i.e. $b_i = v_i$). This time we will apply the VCG mechanism instead of the GSP. The two slots will still be won by a and b, the only difference being what they will be charged for that. As a first step note that, for bidders above bidder i in the ranking, it makes no difference in their payoffs whether i is or is not there. Thus, when computing the negative effect of the presence of a bidder, we must just consider such effect on agents ranked lower than him.

Consider agent b. Had he not been there agent c would have earned the second slot, which would have earned him $100 \cdot 2\$ = 200\$$. So b will get charged exactly 200$, while earning $200 \cdot 4\$$, for a payoff of 600$.

Agent a's absence would have benefited both b and c. As above, c would have won the last slot, which is worth 200$ to him. And b would have gone from the second to the first. In the slot he's in when a participates he earns $100 \cdot 4\$$, while in the first slot he could make $200 \cdot 4\$$. So the loss he has because of player a is $(200 - 100) \cdot 4\$ = 400\$$. This means that agent a will get charged $200\$ + 400\$ = 600\$$, while earning $200 \cdot 5\$$, which gives a payoff of 400$.

Position	Agent	Bid	Price paid	Revenue
1	a	$b_a = v_a = 5\$$	$p_1 = 600\$$	$5\$ \cdot 200 - 600\$ = 400\$$
2	b	$b_b = v_b = 4\$$	$p_2 = 200\$$	$4\$ \cdot 100 - 200\$ = 200\$$
-	c	$b_c = v_c = 2\$$	$p_3 = 0\$$	0$

■

We can now write an explicit formula for the payment p_j^V of slot j under the VCG mechanism:

$$p_j^V = \sum_{k>j} v_k^*(\alpha_{k-1} - \alpha_k), \tag{2.10}$$

assuming as usual that $\alpha_s = 0$ for $s > k$ where k is the number of slots.

Notice that p_j^V represents the overall payment for bidder j: for ease of exposition we will denote by q_j^V his per click payment so that

$$q_j^V = p_j^V / \alpha_j.$$

There are a few interesting facts about the VCG auction mechanism, which we report in the proposition below.

Proposition 2.12.

i) no agent can benefit from misreporting his true valuation: truthful bidding is a dominant strategy (as in the Vickrey second price auction),

ii) VCG prices are non-increasing

iii) the payments computed by the VCG correspond to the buyer optimal minimum equilibrium prices,

iv) there is a GSP equilibrium outcome in which the allocations and prices are the same as in the corresponding VCG outcome,

v) bidder's payments in the GSP are always at least as large as the payments in the corresponding VCG.

vi) VCG prices satisfies local indifference, i.e. for every bidder $i \in [2..k+1]$

$$\alpha_i(v_i - q_i^V) = \alpha_{i+1}(v_i - q_{i+1}^V).$$

vii) VCG prices are envy-free, i.e. for every $i, j \in [1..k+1]$

$$\alpha_j(v_j - q_j^V) \geq \alpha_i(v_j - q_i^V).$$

Proof. Property i) is Theorem 1.5 [page 22].

For ii), consider first that (2.10) can be expressed recursively as

$$\begin{aligned} q_i^V &= \frac{1}{\alpha_i} v_{i+1}(\alpha_i - \alpha_{i+1}) + \frac{1}{\alpha_i} \sum_{j>i+1} v_j(\alpha_{j-1} - \alpha_j) \\ &= v_{i+1}(1 - \frac{\alpha_{j+1}}{\alpha_i}) + \frac{\alpha_{i+1}}{\alpha_i} q_{i+1}^V. \end{aligned} \tag{2.11}$$

Since truth telling is a dominant strategy for VCG, it must be the case that $q_i^V \leq v_i$ (since otherwise the strategy of declaring v_i would be dominated by simply bidding 0 and receiving a 0 payoff). Substituting for v_{i+1} we get $q_i^V \geq q_{i+1}^V$, which also implies $p_i^V \geq p_{i+1}^V$, since we assume that $\alpha_i \geq \alpha_{i+1}$.

The proof of iii) will be outlined in Section 2.6.

To see iv), consider an (SNE) equilibrium for GSP, which satisfies the lower bound (2.8) which we rewrite here:

$$b_j^* \alpha_{j-1} = \sum_{k \geq j} v_{k-1}^* (\alpha_{k-1} - \alpha_k).$$

Since in the GSP (to which the formula above refers) $p^*_j = b^*_{j+1}$, this becomes

$$p^*_{j-1}\alpha_{j-1} = \sum_{k\geq j} v^*_{k-1}(\alpha_{k-1} - \alpha_k).$$

Comparing this with (2.10) we see that the price in the GSP mechanism is

$$p^*_j = \frac{p^V_j}{\alpha_j} = q^V_j .$$

This implies that, at the lowest bids in a symmetric Nash equilibrium, the payments are the same as in the VCG.

The proof of v) is presented below, in Theorem 2.13.

Point vi) follows by simply rearranging (2.11).

Finally to prove part vii) we need to verify the inequalities for slots above and below j. Consider first slots below j's current position, i.e. $i > j$. In this case we know that, by lowering his bid, j could get them precisely at price q^V_i: the envy freeness inequality $\alpha_j(v_j - q^V_j) \geq \alpha_i(v_j - q^V_i)$ is a consequence of the fact that bidding v_i is a dominant strategy in VCG. Now consider slots above j, i.e. $i < j$. By the fact that prices satisfy local indifference, i.e. point vi), for every slot $h \leq j$ we know that $\alpha_h(v_h - q^V_h) = \alpha_{h-1}(v_{h-1} - q^V_{h-1})$. Since $\alpha_{l-1} \geq \alpha_l$ and $v_l \geq v_j$, this implies that $\alpha_h(v_j - q^V_h) = \alpha_{h-1}(v_j - q^V_{h-1})$. Since this holds for all $h \leq i$, it holds also for i and j, so that $\alpha_j(v_j - q^V_j) \geq \alpha_i(v_j - q^V_i)$. □

As a last result we prove what is a very interesting fact in the eyes of the search engine companies selling ad slots: their revenue is always at least as large in the GSP as it is in the VCG (assuming the bidders bid the same amounts). This partially motivates the use of the GSP mechanism, despite the fact that it is prone to strategic manipulations.

Theorem 2.13. *If all bidders were to bid the same amounts under the GSP and VCG mechanisms, then each advertiser's payment would be at least as large under GSP as under VCG.*

Proof. Assume there are more bidders than available slots. Equation (2.10) can be expressed recursively as

$$p^V_j = (\alpha_j - \alpha_{j+1})b^*_{j+1} + p^V_{j+1},$$

and considering the difference between the prices payed by j and $j+1$

$$p^V_j - p^V_{j+1} = (\alpha_j - \alpha_{j+1})b^*_{j+1}.$$

Since $b^*_{j+1} \geq b^*_{j+2}$ we can write

$$(\alpha_j - \alpha_{j+1})b^*_{j+1} \leq \alpha_j b^*_{j+1} - \alpha_{j+1} b^*_{j+2},$$

but the right hand side of the above inequality is precisely the difference of payments from position j to position $j+1$ in the GSP mechanism (call them p^G_j and p^G_{j+1}), so we finally have

$$p^V_j - p^V_{j+1} \leq p^G_j - p^G_{j+1}.$$

□

2.4.1 Generalizations

As in Section 2.3.2 [page 37] we can consider variations of the above mechanism for when click through rates are not bidder independent.

When click through rates are separable, it is possible to use the weighted version of the VCG auction (see Section 1.1.2 [page 24]) to obtain an incentive compatible mechanism in this setting. As noted on page 29, when click through rates are separable, adding bidder specific weights w_i adds nothing to the model, so we will consider all weights set to 1. Total payments can be expressed as

$$p^{wV}_i = \sum_{j>i}^{k+1} \frac{\beta_j}{\beta_i} v_j \left(\alpha_{j-1} - \alpha_j\right), \tag{2.12}$$

so that the per click cost is

$$q^{wV}_i = \frac{1}{\alpha_i} p^{wV}_i. \tag{2.13}$$

As in the GSP case, we can see that this is equivalent to scaling all bids and values by the corresponding β, since agent i's utility is now

$$u_i = \alpha_i \beta_i (v_i - q^{wV}_i) = \alpha_i \left(\beta_i v_i - \frac{1}{\alpha_i} \sum_{j>i} \beta_j v_j (\alpha_{j-1} - \alpha_j) \right). \tag{2.14}$$

On the other hand, if we are given external weights $\{w_i\}_{i=1}^n$, and want the agents to be sorted in decreasing order of $w_i b_i$, we can set the VCG weights to be $\frac{w_i}{\beta_i}$ and get an incentive compatible mechanism.

However when click through rates are not separable there might be rankings (i.e. sets of weights $\{w_i\}_{i=1}^n$) such that there are no corresponding VCG weights that give an incentive compatible mechanism while maintaining the same bidders order. Aggarwal et al. [2006] give a simple example.

Example 2.4.14. Consider a non separable click through rates example in which there are only two slots and two advertisers: a and b. Assume both have a click through rate of 0.4 when in the first slot, while, when displayed in the second slot, a will still receive 0.4 clicks, while b only 0.2. Let $\omega_a, \omega_b > 0$ be the weights assigned to a and b respectively by a weighted VCG mechanism. Since, by definition, such a mechanism maximizes social welfare, we can see when it will rank a above b or *vice versa*. Let b_a, b_b denote a and b's bids respectively. Then b will be ranked above a whenever

$$0.4\omega_b b_b + 0.4\omega_a b_a > 0.4\omega_a b_a + 0.2\omega_b b_b,$$

which is true for all $b_b > 0$. Thus, for any setting of weights, the VCG mechanism will rank b above a. ■

The previous example shows that, when click through rates are not separable, the weighted VCG mechanism might not be incentive compatible. Ashlagi et al. [2009] introduce the laddered auction and show that, in this setting, it is the only incentive compatible mechanism.

2.5 The Efficiency of the GSP

As we have seen in Chapter 1, most of the economics theoretical literature in mechanism design has focused on the development and characterization of truthful mechanism. Why then, one might wonder, in a multi billion dollar application of auctions, have people chosen a non incentive compatible compatible mechanism (GSP), even though a truthful one (VCG) was available?

One of the reasons behind the adoption of GSP auctions lies certainly in the characteristics of its equilibria. In fact, although truthful bidding is not an equilibrium strategy in GSP, there are "good" equilibria, that resemble the VCG outcome. In fact, as already noted, the lower symmetric Nash equilibrium bids (2.8) yield exactly the VCG payments (scaled by the click through rate values). Furthermore, by Fact 2.6, the equilibrium allocation is the same as in the VCG case: social welfare is maximized (this fact is often referred to as **efficiency**). One might ask if this property is a characteristics of the GSP mechanism or if it is shared by other auctions.

Recently Babaioff and Roughgarden [2010] gave a characterization of GSP-like mechanisms in sponsored search, which we briefly outline here. We restrict our attention to a class of mechanisms that are

- **anonymous**: the price an agent pays depend only on the bids (and not on his name),
- **upper-triangular**: the price of bidder in position j depends only on bids strictly below j.

Definition 2.15 (Efficient Mechanisms). *A position auction mechanism is efficient if, for every valuation profile v, there exist a full information Nash equilibrium profile of bids b such that*

i) slots are assigned in order of decreasing valuations (i.e. the efficient allocation in this setting) and

ii) the equilibrium prices are the VCG prices.

Given click through rates α, let $\mathcal{P}(\alpha)$ denote the set of all possible VCG prices, i.e. $\mathcal{P}(\alpha) = \left\{ \frac{1}{\alpha_j} \sum_{j>i} b_j \left(\alpha_{j-1} - \alpha_j \right) \mid v \text{ is a feasible valuation profile} \right\}$. Let $\mathsf{VCG}(v)$ denote the VCG prices when the valuations are v. A final ingredient for the characterization is a monotonicity condition.

Definition 2.16 (Local Monotonicity). *An anonymous and upper-triangular mechanism is locally monotone if its payment rule p is such that, for all valuations v there exist a profile of bids b such that* *i)* $p(b) = VCG(v)$ *(i.e. the prices are the efficient ones),*

ii) $p_{j-1}(b_j, b_{j+1}, \cdots, b_{k+1}) \leq p_{j-1}(b_{j-1}, b_{j+1}, \cdots, b_{k+1})$ *for all* $j \in [2..k+1]$.

We now give the characterization; the proofs can be found in the original paper.

Theorem 2.17. *Fix a click through rate profile* α. *An anonymous upper-triangular mechanism is efficient only if it is locally monotone and for every valuation profile there is a non-negative non-decreasing bid vector such that* $p(b) = VCG(v)$.

The sufficient conditions rely on a different monotonicity condition.

Theorem 2.18. *Fix a click through rateprofile* α. *An anonymous upper-triangular mechanism is efficient if*

- *for any slot j,* $p_j(b') \geq p_j(b)$ *whenever* $b'_i \geq b_i$ *for all* $i > j$,
- *for every valuation profile there is a non-negative non-decreasing bid vector such that* $p(b) = VCG(v)$.

Besides having efficient equilibrium outcomes, the GSP has another characteristic that has often been cited when explaining why it has been chosen over the more sound VCG: simplicity. In fact, in the sponsored search setting, the focus is on creating a simple platform and attract even people that would not usually consider advertising. The VCG payment rule can look daunting, and in any case the GSP is easier to describe (even in the general case): a bidder will pay the minimum amount required to maintain his position.

It can be shown (again, see Babaioff and Roughgarden [2010]) that the GSP minimizes the number of bids the payment depends on. For simplicity we focus on comparing GSP to mechanisms with linear payment rules, i.e. such that p_i has the form $p_i = \sum_{j>i} \lambda_{ij} b_j$. We now define a function that counts the number of dependencies in the payment rule of a mechanism $\mathcal{M}$: $\chi(\mathcal{M}) = \chi^{\mathcal{M}}_{i,j}$. The value of $\chi^{\mathcal{M}}_{i,j}$ is 1 if payment p_i depends on bid b_j and 0 otherwise.

Theorem 2.19. *For every linear payment rule p,* $\chi(p) \geq \chi(GSP)$.

It must also be noted that, although GSP auctions have the guarantee that there is an efficient equilibrium, this is not the only one. In fact the inequalities in (2.7) give a whole range of bids that form symmetric Nash equilibria. Could it be that there are other, highly inefficient equilibrium outcomes? This question is is usually framed in the context of the **price of anarchy**, i.e. the ratio between the lowest social welfare in a Nash equilibrium and the global optimum. Notice that, in the sponsored search case, we already know the value of the optimal social welfare, since it is the one obtained by the VCG mechanism. Leme and Tardos [2009] show that, for full information Nash equilibria, the worst possible bound is $\simeq 1.61$, meaning that even the "worst" equilibrium outcome of GSP is still not far from the optimum.[6]

[6] Their result holds under the weak assumption that agents do not bid more than their valuation (which can be easily seen to be a dominated strategy).

2.6 The Assignment Game

There is a strong connection between multi object auctions and the assignment game first described by Shapley and Shubik in 1972 (Shapley and Shubik [1971]). We will briefly describe this matching problem and then outline its connections to the ad auction problem.

Given a set of agents Q and a set of indivisible objects P let γ_{ij} represent the potential gains from the trade between $i \in Q$ and $j \in P$ (that is if buyer i gets object j). We are interested in determining an assignment of objects to buyers that maximizes the total value. The following linear program captures this optimal assignment,

$$\begin{aligned} \max \quad & \sum_{i,j} \gamma_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i \in Q} x_{ij} \leq 1 \qquad\qquad\qquad\qquad\qquad\qquad (\text{P}) \\ & \sum_{j \in P} x_{ij} \leq 1 \\ & x_{ij} \geq 0 \quad \text{for all} \quad i \in Q, j \in P. \end{aligned}$$

Where $x_{ij} = 1$ if object j is assigned to buyer i. The constraint matrix is totally unimodular, and this guarantees that there is an integral solution to this linear program.

We could interpret the γ_{ij}'s as the true valuation of bidder i with respect to object j. In this case, the program (P) could be seen as maximizing social welfare.

A feasible assignment is a matrix $X = (x_{ij}) \in \{0,1\}$ that satisfies the above conditions, and corresponds to a bipartite matching in which $x_{ij} = 1$ if object j is assigned to i. Given that $x_{ij} \in \{0,1\}$ the constraints to (P) simply state that there is at most one object for each buyer and vice versa.

An assignment X is optimal if for every other feasible assignment X', $\sum_{i,j} \gamma_{ij} x_{ij} \geq \sum_{i,j} \gamma_{ij} x'_{ij}$.

We can assign a payoff to each buyer and each seller. The vectors $u \in \mathbb{R}^n$ and $w \in \mathbb{R}^m$ are a feasible payoff if there exist a feasible assignment X such that

$$\sum_{i \in Q} u_i + \sum_{j \in P} w_j = \sum_{i,j} \gamma_{ij} x_{ij}.$$

Given a feasible payoff and a corresponding feasible assignment we obtain a feasible outcome, which assigns objects to buyers and gives each a payoff. But the payoffs given might not be enough for someone: it could be the case that a buyer could receive a higher payoff by being assigned to another object (an unstable matching). A stable outcome is an outcome in which this is not possible, formally an outcome is stable if

i) $u_i \geq 0, w_i \geq 0,$

ii) $u_i + w_j \geq \gamma_{ij}$ for all (i, j).

The first conditions ensures that nobody is receiving a negative payoff (in which case he would obviously be better off simply not participating). The second condition assures that no pair blocks the assignment X. Assume that this condition is not satisfied for i and j, then they both would be better off breaking their current partnership and forming one together (that would be worth γ_{ij}).

We now consider the dual of (P):

$$\begin{aligned} \text{min} \quad & \sum_i u_i + \sum_j w_j \\ \text{s.t.} \quad & u_i + w_j \geq \gamma_{ij} \quad \text{for all} \quad i, j \\ & u_i \geq 0, w_j \geq 0 \end{aligned} \tag{D}$$

Where u_i are the dual variables relative to the first set of constraints of (P) and w_j are relative to the other set of constraints. Since (P) has an optimal solution, also (D) must have one, and, by the strong duality theorem, we obtain

$$\sum_i u_i + \sum_j w_j = \sum_{i,j} x_{ij}\gamma_{ij}.$$

Thus (u, w) is a feasible payoff for the matching X (and so $((u, w), X)$ is a feasible outcome). The outcome is also stable, since the conditions for (D) are precisely the definition of a stable payoff. It can be shown (see Shapley and Shubik [1971]) that the set of stable outcomes coincides with the core of the assignment game, and that the core itself is just the set of solutions of (D).

An interesting result, shows that there are matchings that are preferred just by one side of the market. Furthermore if in an assignment every agent on one side gets the maximum payoff then every player on the other side will receive the minimum payoff.

Theorem 2.20. *There is a buyer-optimal stable payoff* $(\bar{u}, \bar{w})$ *with the property that for any other stable payoff* (u, w)

$$\begin{aligned}\bar{u} &\geq u \\ \bar{w} &\leq w.\end{aligned}$$

This holds symmetrically for the sellers (objects) side.

2.6.1 Prices

If we consider the two sets P and Q as bidders and objects, we can better describe the situation by introducing prices. The γ_{ij}'s can now be interpreted as the value of object j to bidder i. If we assign a non-negative price p_j to each object j we can define agent's i payoff as $\gamma_{ij} - p_j$. The demand set of bidder i at prices p is the set of objects he is most interested in

$$D_i(p) = \left\{ j \in P \;\middle|\; \gamma_{ij} - p_j = \max_{k \in P} \{\gamma_{ik} - p_k\} \right\}.$$

A price vector p is said quasi-competitive if there is a matching X that matches each agent with an object in his demand set. When p is quasi competitive and μ is the corresponding matching, if $p_j = 0$ for all $j \notin \mu$, then the pair (p, μ) is called a competitive equilibrium. In a competitive equilibrium not only does every agent receive an object he is most interested in (i.e. in his demand set) but any unmatched objects have a 0 price.

If (p, μ) is a competitive equilibrium we can write the payoffs to agents and sellers as

$$\begin{aligned}u_i &= \gamma_{ij} - p_j \\ w_j &= p_j.\end{aligned}$$

It is easy to verify that these definitions give a stable payoff. The existence of a matching that is optimal for the buyers (see Theorem 2.20 [page 49]) corresponds to the existence of a vector of equilibrium prices that is at least as small in every component as any other equilibrium price vector: the minimum equilibrium price.

2.6.2 Connections with the GSP

The assignment model is deeply connected with the GSP mechanism, and in general with all auctions of multiple items (see Bikhcandani and Ostroy [2006]).

In the assignment model, by the definition of equilibrium price and demand set, it follows that if p is an equilibrium price then each agent prefers the object assigned to him over any other object:

$$\gamma_{ij} - p_j \geq \gamma_{ik} - p_k \qquad \text{for all} \quad k.$$

In the ad auction problem the utility of an agent i ($\gamma_{ij} - p_j$ in the assignment model) being assigned to a particular slot j is just the click through rate α_j times the valuation per click of that agent. Substituting such a utility in the above condition we obtain

$$v_i\alpha_j - p_j\alpha_j \geq v_i\alpha_k - p_k\alpha_k,$$

which are precisely the inequalities characterizing the symmetric Nash equilibria. Thus the (SNE) of the ad auction problem with GSP mechanism described above are just the competitive equilibria of a particular matching game. In Edelman et al. [2007] the following theorem is proved.

Theorem 2.21. *If there are more advertisers than slots, the outcome of a GSP auction is a symmetric Nash equilibrium if and only if it is a stable assignment in the corresponding matching game.*

Proof. See Edelman et al. [2007], pp.21-22. □

2.6.3 Connections with VCG

The following theorem gives a strong connection between the VCG mechanism and stable assignments.

Theorem 2.22. *Let $(\hat{u}, \hat{v})$ be the buyer-optimal stable payoff. Then, for all buyers i*

$$\hat{u}_i = v(P, Q) - v(P \setminus \{i\}, Q)$$

where v is a function that, given one set of buyers and one set of objects, gives the maximum possible value attainable, that is

$$v(S, R) = \max \sum_{(i,j)\in S\times R} \alpha_{ij}x_{ij}.$$

Proof. See Roth and Sotomayor [1990], pp. 212-213. □

Intuitively this states that, in the buyers' optimal payoff, each buyer's utility is the difference between the value of the auction with or without him. In particular the term

$$-v(P \setminus \{i\}, Q)$$

is the “harm” buyer i causes to the other participants; what they could have earned if player i were absent. These values correspond to the VCG prices (see 1.1.2 [page 21]), and Theorem 2.22 is a proof of Proposition 2.12, part ii).

2.7 Dynamic Aspects

The cases considered so far deal with a single auction of multiple goods. In the ad auctions scenario this corresponds to the search engine user entering one query, and the ads relevant to that query being displayed alongside the search results in a number of slots.

This modeling is correct, although the situation is more complex. In fact multiple identical queries arrive in a given time period (e.g. a day), so the auctions described above will take place many times. Bidders in ad auctions are usually required to enter a daily budget, the maximum amount of money they are willing to spend in a day. As a query arrives, the search engine assigns the slots according to bid value, and considering only agents with a positive budget.

We introduce a simple model in which there are multiple bidders interested in multiple keywords, each agent has a budget constraint and there is a single slot. To simplify the exposition we will assume that, counter to practice, each bidder is charged his own bid, rather than using the GSP or VCG mechanism. Let m_i be the budget of bidder i, and consider the maximization problem faced by the search engine during a day.

$$\begin{aligned} \max \quad & \sum_{i,j} b_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i \in Q} x_{ij} \leq r_j \\ & \sum_{j \in P} b_{ij} x_{ij} \leq m_i \\ & x_{ij} \geq 0 \quad \text{for all} \quad i \in Q, j \in P. \end{aligned} \tag{S}$$

This corresponds to finding an allocation $X = (x_{ij})$ that maximizes revenue (recall that in the current model each agent pays his bid, so $\sum_{i,j} b_{ij} x_{ij}$ is the revenue of allocation X). The first set of constraints limit the amounts of possible clicks on each keyword, so we can think of r_j as an accurate estimate of the total number of clicks word j will receive. The other constraints are the budget constraints of each agent. The constraint matrix is not unimodular in this case, and so we are not guaranteed that there is an integral solution to (S).

Since the values of r_j are not readily available, (S) is best thought as an offline version of the ad auction problem: given the searches of a whole day, what could have been the maximum revenue by the search engine? On the other hand we could consider an online procedure that assigns queries to bidders (while respecting budget constraints) and maximizes the total revenue. The most obvious solution would be a greed procedure, that simply assigns the slot to the highest bidder with non-negative budget. This approach can yield very poor results, as the next example shows.

Example 2.7.23. There are 2 bidders, *a* and *b*, each of whom has a budget of 2\$. Assume there are 2 queries and only one available slot, and that both agents are interested in both queries:

		Keywords	
		1	2
Agents	a	$2\$$	$(2-\varepsilon)\$$
	b	$2\$$	$\varepsilon\$$

Assume that query 1 arrives, followed by query 2. The greedy algorithm will assign query 1 to bidder *a*, and query 2 to bidder *b*. Note that the budget constraints are preserved. The total revenue for the seller is thus

$$(2+\varepsilon)\$.$$

The optimal solution, instead, would assign query 1 to bidder *b* and query 2 to bidder *a*, obtaining a total revenue of

$$2\$ + (2-\varepsilon)\$$$

almost twice as much as our greedy algorithm. ■

The authors in Kalyanasundaram and Pruhs [2000] give an interesting procedure for the problem of online *b*-matching. This is a simplified version of the problem we are considering, in which each agent has a budget of m but is limited only to 0/1 bids. They define an algorithm that assigns each query to the interested bidder with the highest unspent budget. They then prove that, by using this procedure, the ratio between their final revenue and the optimal value (the competitive ratio) is[7] $(1-\frac{1}{e}) \approx 0.63$.

The authors in Mehta et al. [2007] generalize the results in Kalyanasundaram and Pruhs [2000] to arbitrary bids. The key idea is the same: when assigning the slots keep into account not only bids but also the

[7] Actually the competitive ratio tends to $(1-1/e)$ as $m \to \infty$.

fractions of spent budgets. In a surprising way they determine the following tradeoff function

$$\phi(x) = 1 - e^{-(1-x)}. \tag{2.15}$$

Let m_i be agent i's initial budget, and s_i the amount already spent by i when a query j arrives; the quantity $\phi\left(\frac{s_i}{m_i}\right)$ is computed for every agent, and the algorithm assigns the query to the agent maximizing the product

$$b_{ij}\phi\left(\frac{s_i}{m_i}\right).$$

The competitive ratio of this procedure turns out to be $(1 - \frac{1}{e})$.

Budget Constraints In the linear program (S), we are seeking an optimal allocation subject to the budget constraint $\sum_{j\in P} b_{ij}x_{ij} \geq m_i$. The role of budget constraints in this setting is not so clear-cut. In fact, most advertisers would certainly prefer receiving many more clicks for an ε-increase in their budget, and this scenario is not captured by the above inequalities. Furthermore, in real world ad auctions, agents can change their budgets. Thus the characterization in (S) might be valid only for a time period in which no agent changes his budget.

Part II

Theoretical Results

Chapter 3

Quality Scores[1]

Although the definition of the sponsored search general model (see 2.2 [page 29]) includes the possibility for the search engines to define the ranking by specifying arbitrary weights w_i to be assigned to merchants, usually only two possibilities have been considered:

i) $w_i = 1$ for all i, so that agents are ranked by bid,

ii) $w_i = \mathsf{CTR}_{1,i}$.

This is mainly due to historical reasons, since the first companies in sponsored search used these weight settings (see also Section 2.2 [page 28]). When arbitrary weights have been taken into account (such as in Aggarwal et al. [2006]) the focus has been defining the incentive compatible mechanism in that setting, since VCG might fail to to be incentive compatible for the desired bidders' sorting (see also Section 2.4.1 [page 43]).

The goal of this chapter is to study how equilibrium properties of the GSP auction change when we allow the search engine to arbitrarily specify a ranking.

In the classical sponsored search problem we assume each slot j has a click through rate α_j, and that click through rates decrease with slot position (i.e. $\alpha_1 \geq \cdots \geq \alpha_k \geq \alpha_{k+1} = 0$). When click through rates are separable, each advertiser i is assigned a merchant specific click through rate, β_i. When agent i appears in position i the overall CTR will be

$$\mathsf{CTR}_{i,j} = \alpha_i \beta_j.$$

[1] The work presented in this chapter has already been published as a technical report, see Budinich [2011].

It turns out that, for the two settings of weights considered in the literature, the equilibrium properties are the same as for the simple bidder independent click through rate case, in which slot i gets α_i clicks irrespective of the ad currently being displayed in that position (see Sections 2.3.2 [page 37] and 2.4.1 [page 43]).

3.1 CTR Independent Quality Scores

Although the cases in which $w_i = 1$ or $w_i = \mathsf{CTR}_{1,i}$ are central, in the actual position auctions used by search engines quality scores assigned to bidders are slightly more general. Although they do reflect the ad-specific click through rate, there is also a component to these values that has nothing to do with the number of clicks received[2]. These components could be used by the search engine to have some control over which ads are displayed, since the overall reputation of the sponsored search market is of paramount importance. Consider for example an ad that has a very misleading text (such as a well crafted spam message) and receives lots of clicks. Once the search engine has verified the illicit or dubious nature of the advertisement, it might wish to drive it away from the results page before cheated users stop clicking on ads. Thus we can imagine that weights w_i are the product of a bidder specific click through rate β_i and a value δ_i that is not related to the ad's click through rate and is arbitrarily determined by the search engine.

We wish to study the effects of these δ components of the quality scores. For simplicity we will not consider the click through rate components of the quality scores β, since even in our model, their introduction will be simply equivalent to a rescaling of bids and valuations.

In what follows, unless otherwise noted, with the term "quality scores" we intend just the non-CTR based components, which we call δ_j for advertiser j. Advertisers are sorted by decreasing δb. Again, without loss of generality, we assume that the advertisers are named according to this order, so that advertiser i gets slot i. Payments are, as in Section 2.3.2 [page 37], given by the lowest bid necessary to retain the position, so that

$$p_i = \frac{\delta_{i+1} b_{i+1}}{\delta_i}. \tag{3.1}$$

However utilities are now different, since the quality scores δ do not

[2] See, for instance, http://goo.gl/nRfxi.

affect the CTR:

$$\begin{aligned} u_i &= \alpha_i \left(v_i - p_i \right) \\ &= \alpha_i \left(v_i - \frac{\delta_{i+1} b_{i+1}}{\delta_i} \right). \end{aligned} \tag{3.2}$$

This utility cannot be considered just as a scaled version of the classical model, and, to the best of our knowledge, it has not been studied in the scientific literature.

3.2 Equilibrium

We present some properties of the full information Nash equilibria of this auction, following the presentation in Section 2.3 [page 31]. The results are similar to the ones presented by Varian for the next-price auction, however the quality scores δ play a central role.

As in the usual next-price auction, the equilibrium conditions are different if we consider the player's utility when moving to higher or lower slots. Namely, when a player moves to a slot above him, he has to overbid the agent currently in that slot. If bidder i increases his bid to move up to slot $j < i$ his payment will be

$$p_{i,j} = \frac{\delta_j b_j}{\delta_i}.$$

However, when he moves to a lower slot, he can bid much less that the current occupier's bid: he needs to bid higher than that agent's current price. So, if i lowers his bid to end in slot $k > i$, he will end up paying

$$p_{i,k} = \frac{\delta_{k+1} b_{k+1}}{\delta_i}.$$

Definition 3.1 (Nash Equilibrium). *A Nash equilibrium (NE) set of bids satisfies, for all i,*

$$\alpha_i \left(v_i - \frac{\delta_{i+1} b_{i+1}}{\delta_i} \right) \geq \alpha_j \left(v_i - \frac{\delta_j b_j}{\delta_i} \right) \text{ for all } j < i, \tag{3.3}$$

$$\alpha_i \left(v_i - \frac{\delta_{i+1} b_{i+1}}{\delta_i} \right) \geq \alpha_j \left(v_i - \frac{\delta_{j+1} b_{j+1}}{\delta_i} \right) \text{ for all } j > i. \tag{3.4}$$

Note that we can rewrite (3.3) and (3.4), respectively, as

$$\alpha_i (v_i - p_i) \geq \alpha_j \left(v_i - p_{j-1} \frac{\delta_{j-1}}{\delta_i} \right) \text{ for all } j < i, \tag{3.5}$$

$$\alpha_i (v_i - p_i) \geq \alpha_j \left(v_i - p_j \frac{\delta_j}{\delta_i} \right) \text{ for all } j > i. \tag{3.6}$$

As for the next-price auction setting we consider a particular subset of these equilibria.

Definition 3.2 (Symmetric Nash Equilibrium). *A symmetric Nash equilibrium (SNE) set of bids satisfies, for all i,*

$$\alpha_i \left(v_i - \frac{\delta_{i+1} b_{i+1}}{\delta_i} \right) \geq \alpha_j \left(v_i - \frac{\delta_{j+1} b_{j+1}}{\delta_i} \right) \textit{ for all } j. \tag{3.7}$$

Again, (3.7) can be stated in terms of prices as

$$\alpha_i (v_i - p_i) \geq \alpha_j \left(v_i - \frac{\delta_j}{\delta_i} p_j \right) \text{ for all } j. \tag{3.8}$$

As in the classical case, each agent has a non-negative surplus in a SNE.

Fact 3.3. *In a SNE, $v_i \geq p_i$ for all i.*

Proof. Consider the first non-visible slot, $K + 1$, for which we know that $\alpha_{K+1} = 0$. By (3.8)

$$\alpha_i (v_i - p_i) \geq \alpha_{K+1} \left(v_i - \frac{\delta_{K+1}}{\delta_i} p_{K+1} \right) = 0,$$

and since $\alpha_i \geq 0$ this implies $v_i \geq p_i$. □

Now we consider the equilibrium prices, and show that, when scaled by δ, they are monotonically decreasing.

Fact 3.4. *In a SNE, for all i, $\alpha_{i-1}\delta_{i-1}p_{i-1} \geq \alpha_i \delta_i p_i$. Furthermore $\delta_{i-1}p_{i-1} \geq \delta_i p_i$, and if $v_{i-1} > v_i$ then $\delta_{i-1}p_{i-1} > \alpha_i \delta_i p_i$.*

Proof. Applying (3.8) to slots i and $i - 1$ we get

$$\alpha_i (v_i - p_i) \geq \alpha_{i-1} \left(v_i - \frac{\delta_{i-1}}{\delta_i} p_{i-1} \right),$$

which can be rewritten as

$$\begin{aligned} \alpha_{i-1}\delta_{i-1}p_{i-1} &\geq \delta_i \left(\alpha_i p_i + v_i(\alpha_{i-1} - \alpha_i) \right) \\ &\geq \alpha_i \delta_i p_i, \end{aligned} \tag{3.9}$$

where the second inequality follows from the fact that $v_i(\alpha_{i-1} - \alpha_i) \geq 0$.
From Fact 3.3 we know that $v_i \geq p_i$, so that (3.9) implies

$$\begin{aligned}\alpha_{i-1}\delta_{i-1}p_{i-1} &\geq \delta_i\left(\alpha_i p_i + p_i(\alpha_{i-1} - \alpha_i)\right)\\ &\geq \alpha_i\delta_i p_i + \delta_i p_i(\alpha_{i-1} - \alpha_i)\\ &= \alpha_{i-1}\delta_i p_i,\end{aligned}$$

which completes the proof. □

We can now verify that, indeed, every SNE is a NE.

Fact 3.5 (*SNE* $\subset$ *NE*)**.** *A SNE set of prices is also a NE set of prices.*

Proof. Since, by Fact 3.4, $\delta_{j-1}p_{j-1} \geq \delta_j p_j$,

$$\alpha_j\left(v_i - \frac{\delta_j}{\delta_i}p_j\right) \geq \alpha_j\left(v_i - \frac{\delta_{j-1}}{\delta_i}p_{j-1}\right).$$

Substituting p_{j-1} with (3.1) the right hand side term becomes

$$\alpha_j\left(v_i - \frac{\delta_j b_j}{\delta_i}\right).$$

By the SNE condition and the above inequalities we get that, in SNE,

$$\alpha_i\left(v_i - \frac{\delta_{i+1}b_{i+1}}{\delta_i}\right) \geq \alpha_j\left(v_i - \frac{\delta_{j+1}b_{j+1}}{\delta_i}\right) = \alpha_j\left(v_i - \frac{\delta_j}{\delta_i}p_j\right) \geq \alpha_j\left(v_i - \frac{\delta_j b_j}{\delta_i}\right),$$

which gives exactly (3.3). □

Next we show that, in SNE, the values $\delta \times v$ are in decreasing order.

Fact 3.6. *In SNE, $\delta_i v_i \geq \delta_j v_j$ if and only if $i < j$.*

Proof. Consider the SNE conditions (3.8) for agent i moving to slot j and agent j moving to slot i:

$$\begin{aligned}&i \text{ to } j \quad \alpha_i(v_i - p_i) \geq \alpha_j(v_i - \frac{\delta_j}{\delta_i}p_j)\\ &j \text{ to } i \quad \alpha_j(v_j - p_j) \geq \alpha_i(v_j - \frac{\delta_i}{\delta_j}p_i).\end{aligned}$$

They can be rewritten as

$$\begin{aligned}&i \text{ to } j \quad \delta_i v_i(\alpha_i - \alpha_j) \geq \alpha_i\delta_i p_i - \alpha_j\delta_j p_j\\ &j \text{ to } i \quad \delta_j v_j(\alpha_j - \alpha_i) \geq \alpha_j\delta_j p_j - \alpha_i\delta_i p_i.\end{aligned}$$

Summing these two we get

$$(\alpha_i - \alpha_j)(\delta_i v_i - \delta_j v_j) \geq 0,$$

which shows that $\delta_i v_i$ and $\delta_j v_j$ must be sorted in the same way as α_i and α_j. By our assumptions on CTRs this completes the proof. □

Finally we show that an important property of the classical setting holds here as well: it is sufficient to verify the SNE conditions for one slot above and one below to ensure they hold for all slots. For ease of notation we show this in a setting with 3 slots; we verify that if the conditions are met for slots 1 and 2 and slots 2 and 3, then they are also met for slots 1 and 3.

Fact 3.7. *If a set of bids satisfies* (3.8) *for $i+1$ and $i-1$, then it satisfies them for all j.*

Proof. The SNE conditions for slots 1 and 2 and slots 2 and 3 are:

$$\begin{aligned} &\text{1 to 2} \quad \alpha_1(v_1 - p_1) \geq \alpha_2(v_1 - \frac{\delta_2}{\delta_1} p_2) \\ &\text{2 to 3} \quad \alpha_2(v_2 - p_2) \geq \alpha_3(v_2 - \frac{\delta_3}{\delta_2} p_3), \end{aligned}$$

which we can rewrite as

$$\text{1 to 2} \quad \delta_1 v_1(\alpha_1 - \alpha_2) \geq \alpha_1 \delta_1 p_1 - \alpha_2 \delta_2 p_2 \tag{3.10}$$

$$\text{2 to 3} \quad \delta_2 v_2(\alpha_2 - \alpha_3) \geq \alpha_2 \delta_2 p_2 - \alpha_3 \delta_3 p_3. \tag{3.11}$$

By Fact 3.6 and our assumptions on CTRs we know that $\delta_1 v_1 \geq \delta_2 v_2$, so that (3.11) implies

$$\delta_1 v_1(\alpha_2 - \alpha_3) \geq \alpha_2 \delta_2 p_2 - \alpha_3 \delta_3 p_3. \tag{3.12}$$

Summing (3.10) and (3.12) we get

$$\alpha_1 (v_1 - p_1) \geq \alpha_3 \left(v_1 - \frac{\delta_3}{\delta_1} p_3 \right),$$

which is exactly the SNE condition for slot 1 and 3. The other direction is similar. □

Using these facts it is possible to give a characterization of SNE bids. Since, in SNE, agent in position i does not want to move to slot $i+1$

$$\alpha_i(v_i - p_i) \geq \alpha_{i+1} \left(v_i - p_{i+1} \frac{\delta_{i+1}}{\delta_i} \right).$$

Similarly agent in slot $i+1$ wouldn't prefer slot i, so

$$\alpha_{i+1}(v_{i+1} - p_{i+1}) \geq \alpha_i \left(v_{i+1} - p_i \frac{\delta_i}{\delta_{i+1}} \right).$$

By combining these two inequalities, we obtain that

$$\delta_i v_i \left(\alpha_i - \alpha_{i+1}\right) + \alpha_{i+1}\delta_{i+1}p_{i+1} \geq \alpha_i \delta_i p_i \geq \delta_{i+1} v_{i+1} \left(\alpha_i - \alpha_{i+1}\right) + \alpha_{i+1}\delta_{i+1}p_{i+1}.$$

Switching from prices to bids, and considering slots $i-1$ and i, the above becomes

$$\delta_{i-1} v_{i-1} \left(\alpha_{i-1} - \alpha_i\right) + \alpha_i \delta_{i+1} b_{i+1} \geq \alpha_{i-1}\delta_{i-1} b_i \geq \delta_i v_i \left(\alpha_{i-1} - \alpha_i\right) + \alpha_i \delta_{i+1} p_{i+1}. \tag{3.13}$$

Given that $\alpha_{K+1} = 0$ we can solve for the lower bound recursion, and see that the minimum equilibrium bids are

$$b_i = \frac{1}{\alpha_{i-1}\delta_{i-1}} \sum_{j \geq i}^{K+1} \delta_j v_j \left(\alpha_{j-1} - \alpha_j\right). \tag{3.14}$$

3.2.1 Efficiency

One of the important theoretical characteristics of GSP auctions is that, although it is easy to see that they are not truthful (see Example 2.2 [page 32]) , they always admit a full information Nash equilibrium in which the payments are the same as in the VCG auction (see Section 2.5 [page 45]).

The VCG prices and allocation for this setting were discussed in Section 2.4.1 [page 43]. In particular, when quality scores are δ, the VCG mechanism will optimize the weighted social welfare $\sum_i \delta_i v_i$, and the overall payments will be given by (2.12), which we report here for convenience:

$$p_i^{\delta V} = \sum_{j>i}^{k+1} \frac{\delta_j}{\delta_i} b_j \left(\alpha_{j-1} - \alpha_j\right),$$

along the corresponding per click costs $q_i^{\delta V} = \frac{1}{\alpha_i} p_i^{vW}$. However the overall utility for agent i will be different, since he is only receiving α_i clicks and not $\alpha_i \delta_i$. In particular

$$u_i = \alpha_i \left(v_i - q_i^{\delta V}\right) = \alpha_i \left(v_i - \frac{1}{\alpha_i} \sum_{j>i}^{k+1} \frac{\delta_j}{\delta_i} b_j \left(\alpha_{j-1} - \alpha_j\right) \right).$$

As in the non-weighted case we can prove the following result.

Theorem 3.8. *The GSP auction with arbitrary click through rates admits a full information Nash equilibrium in which the allocation and payments are the same as in the corresponding VCG outcome.*

Proof. Consider the prices associated to the lower bound equations for the SNE, (3.8):

$$p_i = \frac{1}{\alpha_i \delta_i} \sum_{j>i} \delta_j v_j \left(\alpha_{j-1} - \alpha_j\right).$$

These are exactly the per click VCG prices $q_i^{\delta V}$. The VCG allocation is the one that maximizes the weighted social welfare $\sum_i \delta_i v_i$: this corresponds to assigning higher slots to agents with higher weighted value δv. By Fact 3.6 this is precisely what the GSP does. □

Thus most properties of the GSP are preserved in this more general setting. Assuming the search engines choose appropriate values for δ_i, the GSP mechanism will potentially have an efficient outcome. However this poses new questions about the model. Although it is reasonable enough to assume that the click through rates are common knowledge among bidders (see also Section 6.3 [page 108]), this is not so clear for the quality scores δ.

In fact, it is sensible to assume that the bidder independent components of click through rates are in a sense constant: they depend only on the behavior of users. Thus, given enough time, an agent might be able to get an estimate for the click through rates. On the other hand, the click through rate independent quality scores δ are possibly changed very often,[3] making it impossible to estimate them.

Thus, even if core properties of the model are essentially unchanged, the introduction of these quality scores undermines the credibility of the model itself.

3.2.2 An Example

To give an idea of how the search engine might use δ's to affect the ranking consider the following example, in which there is a bidder (which we call 3) that has an very high value, i.e. is willing to make large bids. However, for the sake of this example, we assume he is a spammer, and that the search engine has assigned him a low quality score. Table 3.1 describes the scenario (note that the bidders are named in decreasing δv product). We can compute the minimum SNE bids (using (3.14)), having fixed some CTR values. Table 3.2 shows the bids and CTRs used. The

[3] Again, see http://goo.gl/nRfxi.

i	v_i	δ_i	$\delta_i v_i$
1	4	1	4
2	1	3	3
3	400	0.02	2
4	2	0.5	1

Table 3.1: Values used in our example.

slot	α_i	b_i	p_i
1	500	> 0.76	2.3
2	250	0.76	0.53
3	100	320	200
4	0	2	0

Table 3.2: Equilibrium bids and payments when using quality scores δ.

expected revenue to the search engine in this case is 21283.3.

Given the same bidders and valuations, let's consider now a setting in which all δ's are set to 1, i.e. the classical next-price sponsored search auction. We also compute the SNE lower bound bids for this case (using the same CTRs). Values are reported in Table 3.3. Computing the revenue in this case gives 1900.

i	v_i	b_i	p_i
3	400	> 2.8	2.8
1	4	2.8	1.6
4	2	1.6	1
2	1	1	

Table 3.3: Equilibrium bids and payments for the same bidders in a classical GSP setting.

Finally we can analyze how the revenue changes as the quality score for the spammer, δ_3, changes. The plot in Figure 3.1 compares the revenue as δ_3 increases, keeping all other parameters in the model fixed. As comparison we also show the GSP revenue and total value in the system (i.e. upper bound to the maximum revenue attainable).

The "jumps" in the revenue as δ_3 increases are due to the fact that bidder 3 moves to upper slots as this happens. It is interesting to note

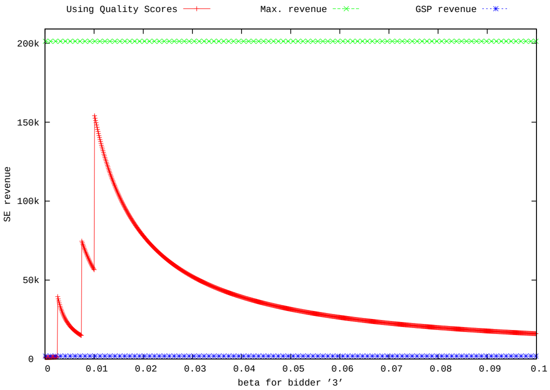


Figure 3.1: SE revenue as the quality score for bidder 3 increases.

that, in Figure 3.1, the maximum revenue is obtained when δ_3 is just enough to put 3 in the first slot. Since all other values are fixed we know that, among other agents, 2 has the highest ranking. We can thus compute the value of δ_3 that maximizes the revenue as the value that solves this equation:

$$\delta_3 b_3 = \delta_2 b_2.$$

Note that, in this case, bidder 3 will pay exactly his bid.

Furthermore, if we fix bids, the maximum revenue the search engine can obtain is by setting δ's so that $\delta_i b_i = \delta_j b_j \forall i, j$. This implies that every bidder is going to pay his bid (which is the maximum possible payment, since the auction rules guarantee that a bidder will never be charged more than his bid).

However this requires that the bids are fixed, or that the quality scores are dynamically computed once the bids arrive. If we assume this is the case, then, to the bidders, this type of auction will be indistinguishable from a first price auction (assuming the search engine breaks ties by giving higher slots to higher bids).

In other words, if we assume that search engines dynamically compute the quality scores (or just update them regularly enough), the whole model collapses, since the payment rule is undefined.

Chapter 4

Strategic Behaviors

The main objective in mechanism design is to devise a set of rules such that, when utility maximizing agents interact following such rules, in equilibrium, the outcome will be the same as the one of a predetermined social choice function. Furthermore, the revelation principle (Theorem 1.2 [page 18]) says that it is sufficient to study "simple" mechanisms, in which all an agent can do is announce a representation of his private information to the auctioneer.

Among all direct mechanism it is possible to characterize all the truthful ones: in this case, in an equilibrium outcome, all an agent has to do is report his actual private information. This greatly simplifies the search for a mechanism (by restricting the class of mechanisms we are looking at), and it has the added benefit that, once we found our mechanism, we are assured that agents will obtain their goal (i.e. maximize their utility) simply by "telling the truth". Thus it would appear that there is little if no space for strategic behaviors, where we loosely define a strategic behavior as the possibility that an agents might benefit by acting differently from what prescribed by the theory.

However strategic behaviors are often observed in real world scenarios and, although very little information is available, also in sponsored search auctions. The main reason behind this is that the models used to study real world auctions are usually (and necessarily) limited. Thus the models for which we can theoretically guarantee strong properties are far from the real world, and for the models that accurately describe the real world it is difficult (if not impossible) to get similar theoretical results.

Another crucial fact in the sponsored search case, is that, even the simplified theoretical model in use by search engines is not incentive compatible, so that agents might be better off not declaring their true value.

4.1 Bidding Rings

A bidding ring is an agreement among a set of bidders prior to the auction itself. In its most classical setting, only one of the bidders will actually participate in the auction. Due to the decreased competition he is likely to pay a lower price; by splitting part of these savings among the other ring participants, the coalition can ensure that every member is better off by participating in the ring. We show that, in sponsored search, bidding rings are not always profitable, and show how quality scores affect these behaviors.

To give an intuition of how bidding rings we consider probably the simplest incentive compatible mechanism: the single item second price auction. We have seen that Vickrey second price auctions and the VCG mechanism are strategy proof when we consider individual bidders, nonetheless they are vulnerable to this kind of manipulation by coalitions.

Example 4.1.1. Consider 5 bidders a, b, c, d, e, whose valuations are

Agent	Value
a	10$
b	8$
c	6$
d	4$
e	1$

In this setting a would win the auction, paying 8$ and thus having a payoff of 2$. Now assume a, b and c decide to form a bidding ring. To this end they decide to submit just 1 bid[1], the highest one. We can represent them as a new bidder r, whose valuation is the same as a's. The situation is now

[1] We can think that they conduce a second price auction between them, in which the good to be sold is the possibility to participate in the original auction.

Agent	Value
r	10$
d	4$
e	1$

The winner will now be *r*, that will pay a price of 4$, thus earning 6$. For *d* and *e*, who do not participate in the ring, nothing has changed. But all the participants in the ring are strictly better off in this scenario. Assuming that *a* will get to keep the object, she can devolve 3$ to *b* and *c*. So the payoffs in the ring are (compared to the payoffs in the case where there is no ring)

Agent	Payoff with Ring	Payoff without Ring
a	3$	2$
b	1.5$	0$
c	1.5$	0$
d	0$	0$
e	0$	0$

■

Clearly there can be many other ways to divide the earnings among the ring participants. The above example just shows that, in general, bidding rings are profitable for participants, and indifferent for outside players. The only negatively affected part is the seller: in the example above his earnings drop from 8$ to 4$.

In real world auctions bidding rings are almost always illegal, but can be very difficult to detect. The ad auction setting poses some new problems in this sense, since identities are a vague concept on the Internet. On the other hand, the actual monetary value of a click to a bidder is not clear; it questionable whether a bidder would like to give up his slot in favor of a pecuniary compensation.

In sponsored search auctions it is conceivable that bidding rings might be very easy to form: consider for instance a company that is in charge of advertising for many firms (Ashlagi et al. [2009] consider such possibility, albeit without side payments), or an advertiser that ensures prominent advertising space on his landing page for the other ring participants.

For a ring to be profitable it must be the case that the increased utility for the bidder that will actually make the bid (call him *a*) must be more than the loss by the bidders that do not participate anymore in the auction. When this happens, *a* will be able to pay the other ring participants.

For simplicity, throughout the rest of this section, we consider a 3-slot 4-bidders setting.

4.1.1 Bidder Independent Click Through Rates

We show how, in the sponsored search setting, bidding rings might not be profitable. We first consider the classical setting, in which all weights are set to 1.

Fact 4.2. *There exist settings of click through rates and values that make bidding rings not profitable using the GSP mechanism.*

Proof. We give an example where, for simplicity, we consider a setting with three slots and four bidders. Assume bidder 1 wants to form a ring with bidder 2. Currently bidder 2's utility is

$$\begin{aligned} u_2 &= \alpha_2(v_2 - p_2) \\ &= \alpha_2(v_2 - b_3) \\ &= \alpha_2\left(v_2 - \frac{v_3(\alpha_2 - \alpha_3) + v_4\alpha_3}{\alpha_2}\right), \end{aligned} \tag{4.1}$$

where the last inequality follows from (3.14) with $\delta_i = 1$. Similarly, bidder 1's utility is

$$u_1 = \alpha_1\left(v_1 - \frac{v_2(\alpha_1 - \alpha_2) + v_3(\alpha_2 - \alpha_3) + v_4\alpha_3}{\alpha_1}\right). \tag{4.2}$$

After the ring has formed, bidder 2 will get paid not to participate, and the new scenario is described in Table 4.1. Since 2 is not participating

slot	CTR	bidder i	b_i
1	α_1	1	b'_1
2	α_2	3	b'_3
3	α_3	4	b'_4
4	$\alpha_4 = 0$	-	-

Table 4.1: The auction after bidders 1 and 2 formed a ring. Notice that bidder 2 is absent.

his utility from the auction will be 0, while we can compute the new SNE and see that 1's utility will now be

$$\begin{aligned} u_1' &= \alpha_1(v_1 - p_2') \\ &= \alpha_1(v_1 - b_3') \\ &= \alpha_1 \left(v_1 - \frac{v_3(\alpha_1 - \alpha_2) + v_4(\alpha_2 - \alpha_3)}{\alpha_1} \right). \end{aligned} \tag{4.3}$$

Since $v_2 \geq v_3 \geq v_4$ and $\alpha_3 v_4 \geq 0$, 1 will profit from the non-participation of 2, i.e. $u_1' \geq u_1$. However, is this increased profit enough to pay 2? Formally, is $u_1' - u_1 \geq u_2$? By substituting and simplifying, we can rewrite $u_1' - u_1 - u_2$ as

$$\alpha_1(v_2 - v_3) - \alpha_2(2v_2 - 3v_3 + v_4) - \alpha_3(2v_3 + 3v_4).$$

First notice that this quantity might be negative, i.e. the ring might not be profitable. Fix any v_1 and v_2, and let $v_3 \simeq v_4 \simeq v_2$, so that the above becomes

$$\simeq 0 - 0 - 5v_2\alpha_3 \leq 0.$$

To see that, on the other hand, in some cases the ring might be profitable, just set $\alpha_1 = 4, \alpha_2 = 2, \alpha_3 = 1$, so that the above becomes simply $v_4 \geq 0$. □

An interesting aside is that, in our examples of profitable vs. non-profitable ring, both conditions hold for *any* $v_1, v_2 \geq 0$. So that, for the two bidders participating in the ring, the profitability of their collusion depends on the external environment (i.e. the values of other bidders and the click through rates).

Finally we note that the same reasoning (with the same conclusions), can be carried out for the case in which bidder 2 is proposing the ring. In this case, the condition for the ring to be profitable will be that $u_2' - u_2 \geq u_1$.

4.1.2 Arbitrary Quality Scores

We now consider the model with bidder independent click through rates and arbitrary rankings (as discussed in Chapter 3 [page 57]). Even in this setting, as in the one described in the previous section, it is not always true that bidding rings are profitable. To see this just notice that the previous model is a special case of the weighted one. Despite there being no apparent difference with what described above, there is an interesting scenario that could greatly increase the possible revenue of a bidding ring.

Assume there is a bidder H with a very high quality score (i.e. weight) but a low budget, while another merchant S (which we might think of as a spammer) with a low quality score but huge budget. In the case these two decide to from a ring, it is conceivable that they could decide to use H's account (and text ad), thus utilizing his quality score, and S's bid and budget, potentially gaining much more as compared to the case in which they have the same weight.

To do this H's ad must simply redirect to S's page. Now S might decide to pay H for this service, or even to have H's ad prominently displayed on S's page. In this case H, although receiving many less clicks, would get them at zero cost. Even if this scenario poses some issues, and surely there are times in which the above methodology won't be applicable (for instance merchants highly connected to their brand names)[2] it is nonetheless an interesting possibility.

This idea could be further exploited, and becomes indeed more plausible, by agents who participate in the market with the only aim of accumulating a high quality score, thereafter selling the account (and the connected quality score) to other merchants.

However the amount by which this scheme might be profitable depends strictly on the actual values of the weights, bids and click through rates. To give an idea of this possibility we consider the numerical values in Section 3.2.2: In this case, the ring between bidders 2 and 3 (where 3 represent a possible spammer) is always profitable. By the preceding discussion they will appear as a bidder $3'$, with value $v_{3'} = v_3 = 400$ and quality score $\delta_{3'} = \delta_2 = 3$. Keeping all other parameters fixed, as the quality score of bidder 3, i.e. δ_3, changes we can see how this affects the profitability (i.e. the excess revenue $u'_3 - u_3 - u_2$) (see Figure 4.1). Notice that the new bidder $3'$ will appear in different slots, according to the value of δ_3.

i	v_i	δ_i	$\delta_i v_i$
1	4	1	4
2	1	3	3
3	400	0.02	2
4	2	0.5	1

[2] As an example, Nike might not want to display Adidas' ads on their home page.

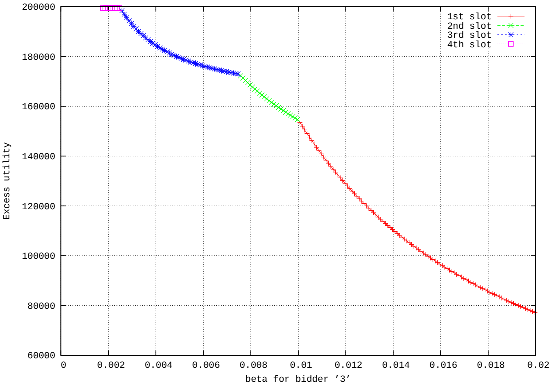


Figure 4.1: Excess revenue of a 2-3 ring as the quality score for bidder 3 increases.

4.2 The Starvation Strategy

Another strategic behavior that might be profitable in ad auctions is based on the fact that each auction is repeated many times. Thus, although each single iteration can be theoretical made strategy proof (coalitions aside), there are profitable deviations for individual agents from truthful behaviors in the long run.

Example 4.2.3. Again, for simplicity, consider only one slot and a Vickrey second price mechanism. This time there are just 2 bidders *a* and *b* with valuations

Agent	Value
a	10$
b	2$

As long as *a*'s budget is not finished there is no chance for *b* to win the slot. But, assuming there are no more competitors, *b* can accelerate this process by bidding more than his true valuation. In this way he increases the amount *a* has to pay for each click, and *a*'s budget will get rapidly exhausted. In fact *b*, by a simple trial and error process, can bid just below *a*'s bid (and, being this a second price auction this coincides with *a*'s valuation). Once *a*'s budget is over then *b* will immediately lower his increased bid to his true valuation, and will then obtain the slot. ■

It is interesting to note that, in some cases, also the seller can benefit from this type of behavior, since *a*'s payments will be higher. This strategic behavior is inherently connected to the dynamic model of the sponsored search setting (see Section 2.7 [page 53]), and we are currently working on framing it theoretically. However we have investigated it experimentally, and results are described in Section 6.2 [page 105], which show that it is not so obviously profitable as the above example might have one believe.

One of the unrealistic points of this strategy is that it relies heavily on budgets. As already described in Section 2.7 [page 53], their role in position auctions is ambiguous. Although it is indeed true that bidders can specify a spending limit, the fact that they can increase (or decrease) their budget very often, makes this limit less of a strict bound. Therefore strategies and ranking functions based on hard budgets seem to have less connections to real world sponsored search auctions.

Part III

Experimental Results

Chapter 5

A Sponsored Search Simulator

As we have seen, today most of the on-line advertising is controlled by search engines and, in particular, sold and delivered using sponsored search auctions (see Chapter 2 [page 27]). There are huge economical interests in these markets both from the sellers (i.e. search engines) and from the advertisers. One of the ways in which search engines try to protect these interest is by hiding many details regarding the inner working of these auctions.

To gain more insight on the workings of these markets, we developed a large scale simulator of sponsored search auctions. The main focus of this simulator is to study and understand the large scale phenomena that occur in these markets, and how the behaviors of groups of agents (i.e. advertisers) affect it. One of the overall design principles has been to keep the environment as simple as possible, in order to be able to focus on the effects of specific behaviors, i.e. maximizing the likelihood that the observed responses were a consequences of the changes made in the system.

Thus, as opposed to other simulations[1], in which most of the focus is on the implementation of "smart" and complex agents, our approach was to have very simple agents. Another important characteristic is that we focus on a comparatively short time frame: most of the simulations run for less than five million queries. Again this is to minimize complexity, focusing on precise events and their effects on the market.

[1] Such as the Trading Agent Competition, in which different agents compete in a simulated market to see who gets the highest revenue. See `http://goo.gl/PcLfg`.

In this chapter we present the working principles of our simulator and of its implementation. We also outline the algorithms and methodologies used to harvest the limited available data in order to obtain reasonable estimates for most of the simulators parameters.

Up to now, the models presented have been a reasonable theoretical representation of the actual auctions in use by all major search engines. At this point, however, we will restrict our attention to a search engine in particular: Google. Although our simulation results and techniques apply to other search engines' markets, when collecting the data we focused on Google's search engine. The main reason for this choice was the presence of some convenient tools that Google developed for its sponsored search program and that allowed us to collect valuable data.

5.1 Data Collection

The starting point in designing the simulator was the collection of some publicly available data on ad auctions. We found some precious resources, which allowed us to collect the following information:

- a reasonable set of words,
- an estimate of the cost of each word,
- an estimate of the number of clicks received by each word.

5.1.1 The Word List

The simulator uses a finite set of words; these words represent all the possible queries that a user can submit to the search engine and also all the possible keywords an advertiser can bid on. The core of the word list has been taken from the SCOWL[2] project (an open source project that maintains a set of word lists for use by spell checkers), and consists of 35867 entries. In a successive development phase we expanded the word list con contain multi term queries and keywords, the details are presented in Section 6.3.3 [page 111].

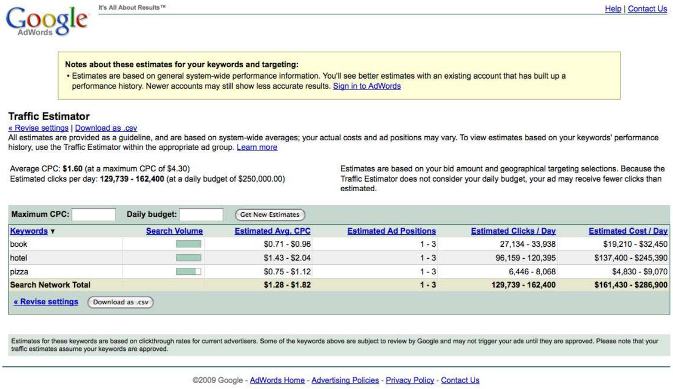
5.1.2 The Traffic Estimator

Google has an on-line tool (the AdWords Traffic Estimator Sandbox,[3]) developed to aid advertisers in their campaigns. The Traffic Estimator,

[2] `http://wordlist.sourceforge.net/`

[3] `https://adwords.google.com/select/TrafficEstimatorSandbox`

given a keyword, displays its estimated cost per click (CPC) and the estimated number of clicks per day (Figure 5.1). The simulator uses this data to estimate some quantities that would otherwise be difficult to generate realistically. Although, as Google itself warns, the data is to be considered only as a guideline, it is of great help for our purposes.



Notes about these estimates for your keywords and targeting:

- Estimates are based on general system-wide performance information. You'll see better estimates with an existing account that has built up a performance history. Newer accounts may still show less accurate results. Sign in to AdWords

Traffic Estimator

« Revise settings | Download as .csv

All estimates are provided as a guideline, and are based on system-wide averages; your actual costs and ad positions may vary. To view estimates based on your keywords' performance history, use the Traffic Estimator within the appropriate ad group. Learn more

Average CPC: **$1.60** (at a maximum CPC of $4.30)
Estimated clicks per day: **129,739 - 162,400** (at a daily budget of $250,000.00)

Estimates are based on your bid amount and geographical targeting selections. Because the Traffic Estimator does not consider your daily budget, your ad may receive fewer clicks than estimated.

Maximum CPC: **Daily budget:**

Keywords ▾	Search Volume	Estimated Avg. CPC	Estimated Ad Positions	Estimated Clicks / Day	Estimated Cost / Day
book		$0.71 - $0.96	1 - 3	27,134 - 33,938	$19,210 - $32,450
hotel		$1.43 - $2.04	1 - 3	96,159 - 120,395	$137,400 - $245,390
pizza		$0.75 - $1.12	1 - 3	6,446 - 8,068	$4,830 - $9,070
Search Network Total		**$1.28 - $1.82**	**1 - 3**	**129,739 - 162,400**	**$161,430 - $286,900**

Estimates for these keywords are based on clickthrough rates for current advertisers. Some of the keywords above are subject to review by Google and may not trigger your ads until they are approved. Please note that your traffic estimates assume your keywords are approved.

©2009 Google - AdWords Home - Advertising Policies - Privacy Policy - Contact Us

Figure 5.1: A screenshot of the website hosting the Traffic Estimator, from which data regarding queries was collected.

The estimated CPC is used in the simulator (averaging the two values given by the Traffic Estimator) as a basis to assign a “real” value to each keyword. The simulator successively employs these values as parameters to generate the agents’ bids and valuations. Clearly the estimated CPC of a term is different from its “real” value. If we were to measure the estimated CPCs in the simulator at the end of a run they would certainly be different from the ones supplied by the Traffic Estimator. Nonetheless their distributions and main features would be similar, and that is enough for the use we make of it.

The other values which are central to the simulator are the estimated number of clicks per day. Since the simulation considers only the queries that give rise to a click, we can simply consider the estimated number of clicks per day as the distribution of the queries in the simulator.

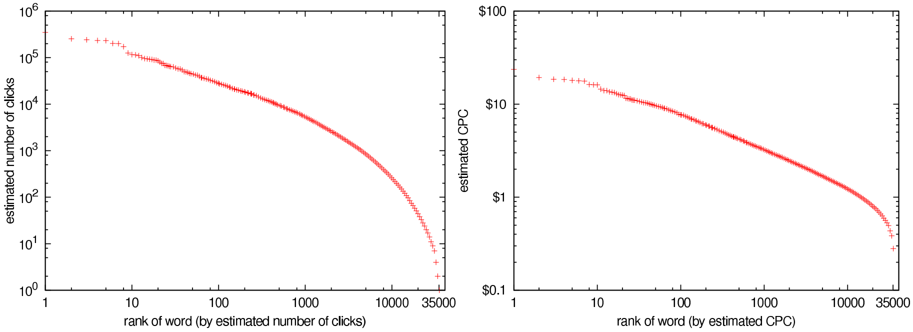
We collected such data for each of the 35867 entries in our dictionary, building a small database that constitutes our initial data set.

5.2 Analysis of the Data Set

In an initial phase of development, the data set has been analyzed. Table 5.1 summarizes its the main characteristics. For the sake of completeness, we plotted the data collected from Google's Traffic Estimator. Figure 5.2a is the distribution of the estimated clicks per day, while Figure 5.2b shows how estimated average costs per click are distributed.

Number of words	35867
Max. clicks per day	349216
Min. clicks per day	1
Max. CPC	\$23.6
Min. CPC	\$0.05

Table 5.1: Some figures from the data set.



(a) estimated number of clicks

(b) estimated CPC

Figure 5.2: Data gathered from Google's Traffic Estimator, in log-log scale.

5.2.1 Correlation in the Data Set

Starting from a list of words, we have expanded it with information on prices and number of clicks. It seems now a natural question to ask if there is any correlation between these quantities. As a first guess it might seem reasonable to expect at least some correlation. That is, we might expect that some "popular" words receive many clicks and have a high price.

Somewhat surprisingly, a superficial analysis gives a negative result. At a first glance the data set exhibits virtually no correlation between the different values.

Figure 5.3 ranks words by estimated number of clicks, and shows these values along the CPCs (both normalized). It looks like there is no order in the CPC values; they appear as if uniformly distributed. To quantify this observation we also measured the Kendall τ rank correlation coefficient, which is 0.1285. Further analysis should be carried out to confirm these impressions, keeping in mind that these datasets should be considered only as rough estimates of the quantities they describe.

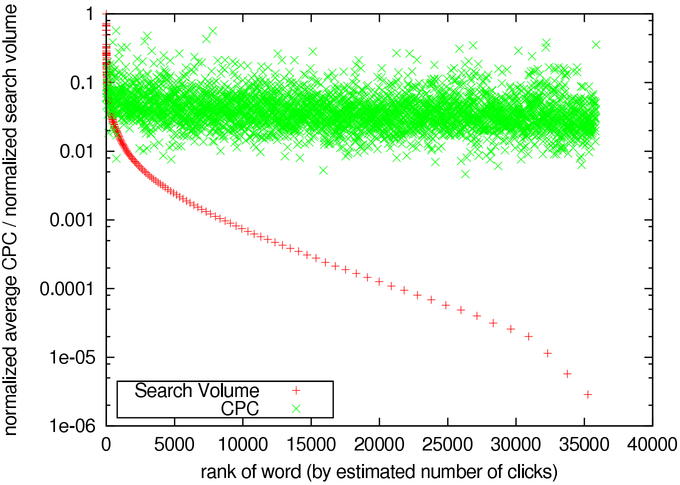


Figure 5.3: Data ranked according to estimated number of clicks while both this value and the estimated cost per click are shown (in log-scale).

5.3 The Simulator

The simulator has a very simple structure (see Figure 5.4). It keeps the agents and the words stored in two dynamic arrays. Each word contains a pointer to a balanced binary search tree that contains the bids on that keyword. Along the bids, each node of the tree stores a pointer to the agent that placed that bid.

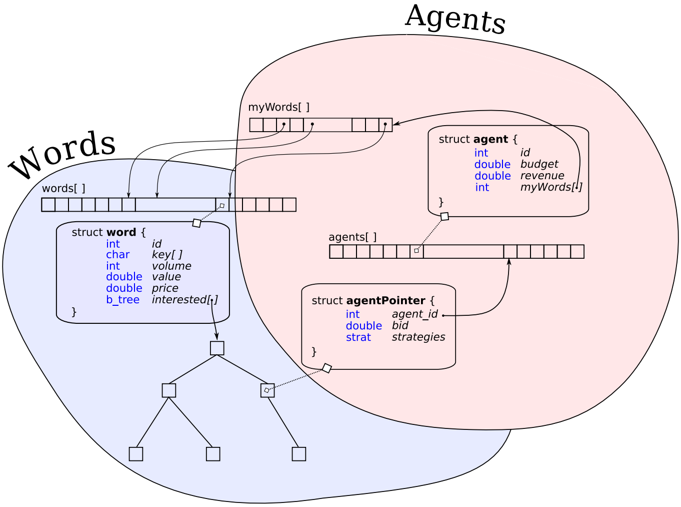


Figure 5.4: Outline of the simulator's data structures.

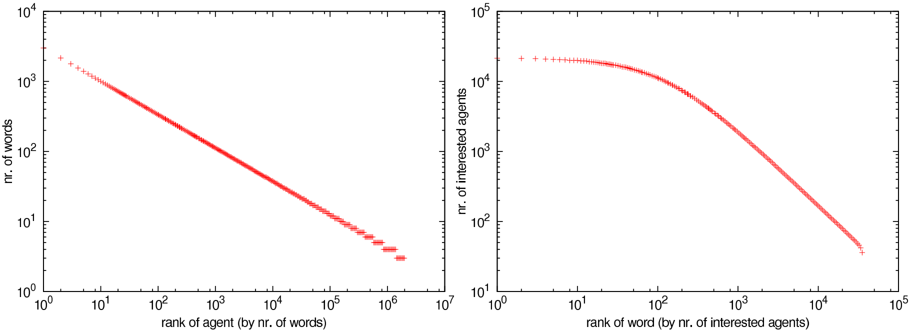
5.3.1 Experimental Details

All the simulations described in this document were carried out using the same set of agents. To this end, the set of agents was generated once and for all and saved in a file. Its main characteristics are presented in Table 5.2. In what follows we will refer to this fixed set of agents and words as our data set.

Nr. of agents	$2 \cdot 10^6$
Max. interested agents in a single word	21446
Min. interested agents in a single word	21
Max. nr. of for an agent	3000
Min. nr. of keywords for an agent	3
Budget Range	[\$1 – \$100]
Bid Range	[\$0.01 – \$200]
Nr. of slots	4
Click through probabilities	0.6, 0.25, 0.10, 0.05

Table 5.2: Some figures from the generated agents (gathered using a sample run with 2×10^6 agents).

Agents Each agent bids on a number of different keywords. The number of different keywords he is going to bid on is randomly drawn from a power law distribution. Once this value has been fixed, keywords are randomly chosen until the required amount is reached. If we consider all the agents, the number of keywords per agent is distributed as a power law, whose parameters are based on the number of agents, such as to keep a fixed maximum and minimum (to avoid cases in which an agent bids on all of the words, or cases in which there are agents that haven't bid on any word at all). Figure 5.5a plots these values for the data set. Another quantity characterizing agents is their budget. In this case a uniform distribution has been chosen, and the budgets are all in the [\$1 − \$100] range.



(a) words per agent **(b)** agents per word

Figure 5.5: Agents per word and words per agent (in log-log scale).

Words Having fixed the number of words an agent will bid on, the next step is to select them from the dictionary. The simulator does so, and the resulting values (i.e. the number of agents interested in every word) is again distributed as a (different) power law. The parameters controlling such distribution are chosen as to avoid unrealistic scenarios. Figure 5.5b shows the number of interested agents per word in our data set.

As described in Section 5.1 each word is assigned a "real" value based on the data gathered from the Traffic Estimator. Based on this reference value, each agent i will then compute its personal valuation v_i for the keyword. The distribution of the valuations for each agent is a normal distribution whose mean is precisely the "real" value of the word. To increase the variety among agents, each agent has a different variance associated to this normal distribution. Figure 5.6 shows the the distribution of valuations for different agents interested in the same keyword (i.e. "reviews").

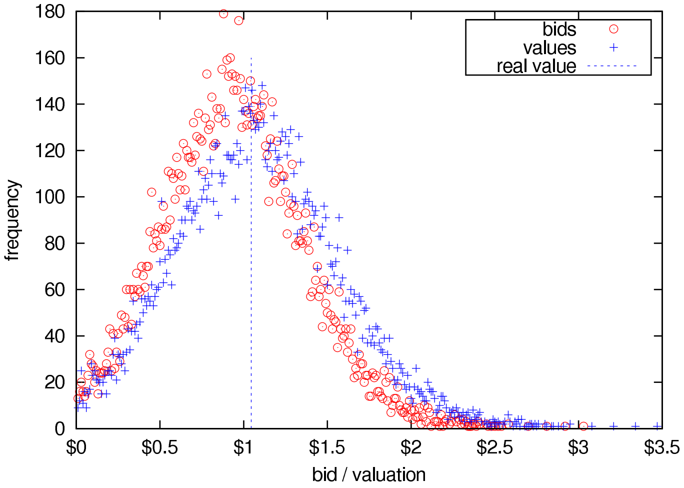


Figure 5.6: The bids and valuations for a single word (i.e. "review"), whose actual value is 1.045.

As a final step each agent i must generate a bid b_i. Bids are generated according to the agent's valuation v_i. Only bids such that $b_i < v_i$ are considered, and they are generated so that the differences $v_i - b_i$ are distributed according to a power law. Figure 5.7 shows the resulting plot for the keyword "reviews".

Keyword	"reviews"
"Real" value	$1.045
Estimated nr. of clicks	12029
Nr. of interested agents	11023
Max. bid	$3.064
Min. bid	$0.010
Max. difference $v_i - b_i$	12.5% of v_i

Table 5.3: Details about a sample keyword and all the agents interested in it.

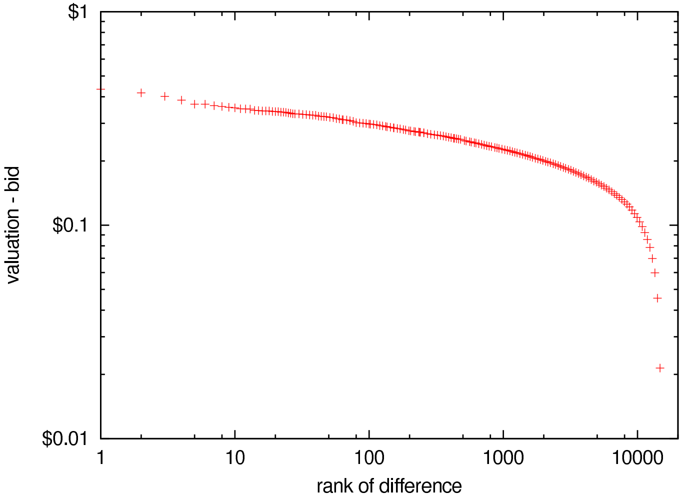


Figure 5.7: Difference between valuations and bids for word “reviews”.

Except for the maximum number of interested agents per word, all the other characteristics described are parameters to the simulator. The number of interested agent per word is indirectly controlled by setting the maximum and minimum number of words a single agent can bid on.[4]

5.4 First Experimental Evaluations

All simulations described were carried out on a standard PC, running Xubuntu (Linux Kernel 2.6.28). The CPU is an Intel Core 2 Duo @2.2GHz, and the system in equipped with 4GB DDR2 667MHz main memory. Some sample running times are reported in Table 5.4. The simulations where performed with the default data set, for a total of $3 \cdot 10^6$ queries. The simulator's greatest bottleneck was the memory usage, which for this scenario was around 2GB.

The simulator implements both the GSP and VCG mechanism, so one of the first objectives was to verify its behavior in known conditions. Thus we compared the revenues with the same agents (and the same bids) under the two mechanisms (see Theorem 2.13).

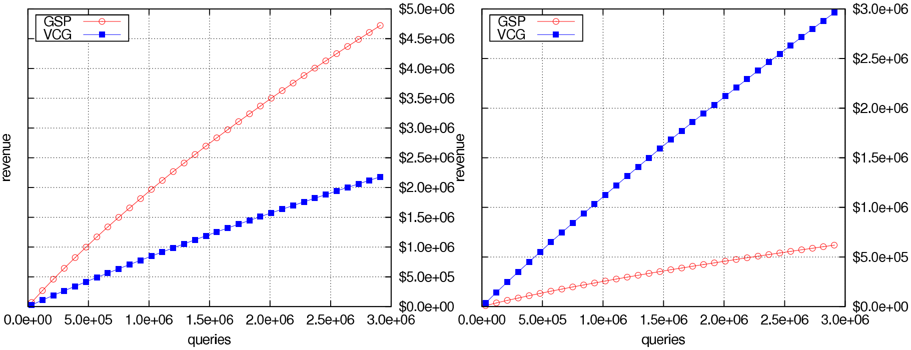
[4] More detailed parameters regarding the distribution of keywords to agents can be easily set.

Simulation Type	Running Time[a]
GSP, with agent generation	1m16s
GSP, agents read from file	1m5s
VCG, agents read from file	1m4s

[a] As reported by the `GNU time` utility, considering only the `user` time.

Table 5.4: Running times of a simulation in standard conditions.

Below (Figure 5.8) is the data gathered in a single run. Each plot displays the data for two repetitions of an identical simulation, one using the GSP mechanism and the other one using the VCG mechanism. The graphs refer to the data recorded by the simulator. Figures 5.8a and 5.8b show the total revenue accrued by the search engine and the agents, respectively.



(a) search engine's revenue

(b) agents' revenue

Figure 5.8: A sample run with the standard data set and for a total of 3×10^6 queries using the GSP and the VCG mechanism.

As predicted by Theorem 2.13 [page 42], the revenue for the search engine is higher using the GSP mechanism. In turn the revenue for the agents is lower under the GSP.

5.4.1 Best-Response Bidding

Cary et al. [2008] present an adaptive bidding strategy and show that, if all bidders employ it, it leads to a VCG like equilibrium in a GSP auction (as in Section 2.5 [page 45]). During the first experimental runs of the simulator we also implemented this bidding strategy and compared its revenue with the one in which bids are distributed as described in Section 5.3.1. For the sake of completeness we briefly outline the balanced bidding strategy and its properties (for more details and proofs see Cary et al. [2008]).

The **balanced bidding** strategy is defined as follows. At each round player i, given as input the other bids b_{-i}, will compute

i) the optimal slot s^* at the current prices, i.e.

$$s_i^* = \arg\max_{s \in \text{slots}} \left\{ \alpha_s(v_i - p_j(s) \right\},$$

where $p_i(s)$ is the price i would pay if he bid as to win position s, and

ii) his bid for the next round b, setting it so that

$$\alpha_k \left(v_i - p_k(i) \right) = \alpha_{k-1} \left(v_i - b \right),$$

where k is i's optimal target slot, i.e. $k = s_i^*$. Thus

$$b = \frac{\alpha_k \left(v_i - p_k(i) \right) - \alpha_{k-1} v_i}{\alpha_{k-1}}.$$

If $s_i^* = 1$ then the above is undefined, so we arbitrarily set $b = \frac{v_i + p_1(i)}{2}$.

This strategy is targeting the slot that will maximize i's utility, and picking a bid b such that i is indifferent between getting the optimal slot and being forced to the slot above that (this could happen by the bidder above i lowering his bid). It can be shown that if all bidders follow the balanced bidding strategy, the system has a unique fixed point that corresponds to the VCG-like equilibrium of GSP.

It can also be shown that the balanced bidding strategy converges, however only in the case in which bidders update their bids one at a time (i.e. asynchronously). However, by employing a slightly different strategy, we can get convergence even in the synchronous case. This strategy is called **restricted balanced bidding**, and is essentially a version of balanced bidding in which agent i is restricted to picking the optimal slot *below* his current position.

We implemented and tested both the restricted and non-restricted version of balanced bidding. Figure 5.9 shows the convergence of the balanced bidding strategy for a single keyword, namely "jobs". In partic-

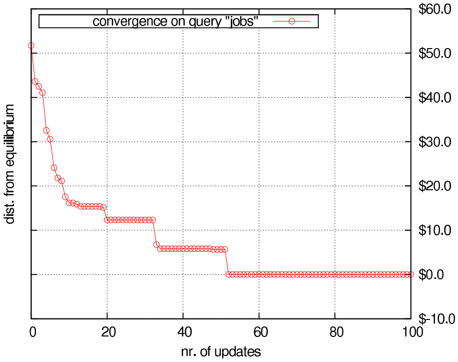
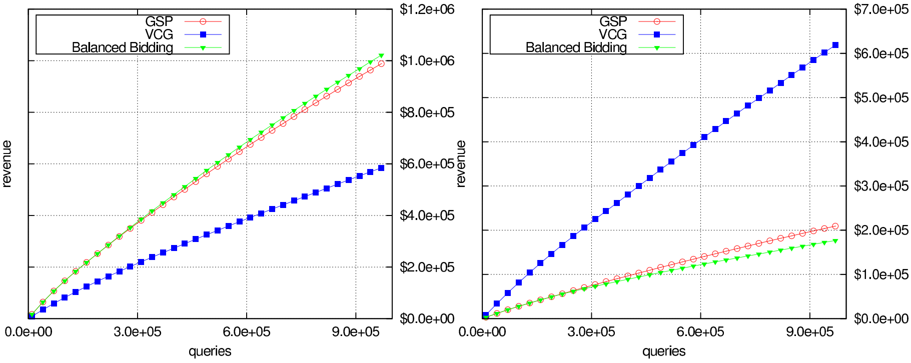


Figure 5.9: Distance from the equilibrium bids when using the balanced bidding update rule.

ular it shows the average bid distance from the computed equilibrium bids as successive queries, and thus bid updates, are performed. As already verified by Cary et al. [2008], convergence is achieved in very few iterations. We also ran some experiments (see Figure 5.10) in which every agent updates his bid using the balanced bidding strategy. Figure 5.10 compares this setup with a VCG and GSP one in which, instead, bidders hold their bids fixed. To highlight the changes when agents start implementing this adaptive bidding strategy, the first 20% of queries are always made with fixed bids.

Given the extremely fast convergence (see Figure 5.9), the global effects when considering millions of queries is small. For this reason, also considering that updating bids at each query affects the simulator's performance, in what follows we will always consider fixed bids.



(a) search engine's revenue

(b) agents' revenue

Figure 5.10: We compare the revenue of the GSP mechanism in which all agents use the balanced bidding strategy (after an initial portion of plain queries) to the VCG and GSP auction without this adaptive behavior.

Chapter 6

Experimental Results

In this chapter we present some experimental results obtained using the simulator described in Chapter 5 [page 77]. The main objective of the simulator is to analyze what happens in these large markets when different fractions of agents engage in new behaviors. We will first see what happens when agents decide to optimize the set of keywords they are currently bidding on. Successively we will investigate the starvation strategic behavior introduced in Section 4.2 [page 74]. Finally we will deal with the problem of data collection, outlining a procedure to estimate an ad's click through rate using only publicly available data.

6.1 Keyword Spreading[1]

A central problem in AdWord markets from the point of view of a seller of goods and services is the generation of keywords. Advertisers typically prefer to bid for keywords that have high search volumes; however they may be very expensive, so that it might be reasonable to bid instead for several related and low volume, inexpensive terms that generate roughly the same amount of traffic altogether. Some preliminary work exploring this idea has been done in Abhishek and Hosanagar [2007], where however, the emphasis is on the algorithmic aspects of keyword generation, not on the global market phenomena as in the present work. A problem related to *keyword spreading* is that of *keyword selection* (see Rusmevichientong and Williamson [2006]), where the economic players try to select at fixed rounds the subset of keywords that

[1] The work presented in this section has already appeared in publication, see Budinich et al. [2010a].

maximize revenues while trying to learn basic parameters (such as keyword click-through rates) during the repeated bidding processes. Note that here the viewpoint is that of a single player and that the the market, as seen by the seller, is modeled via (known or unknown) time varying probability distributions. In contrast, in our simulations keywords are selected by the agents off-line. We simulate directly the market and the auctions by using a large number of atomic agents each performing simple actions. Previous research on agent-based simulation of AdWords markets by Mizuta and Steiglitz [2000] was centered on studying the interaction of different classes of players according to their bidding time profiles, e.g. early vs late bidders. Kitts and LeBlanc [2004] describe a large scale simulator for AdWords markets to investigate several bidding strategies, e.g. random bidding vs. bid to keep relative position, which however do not involve keyword spreading. To the best of our knowledge this is the first large-scale agent-based simulation of the market effects of keyword spreading.

Ad auctions are a peculiar type of auction, since the buyers are not directly interested in the product being sold (i.e. the clicks). Clearly it's not important, from an advertiser's point of view, "where" the clicks come from. What matters is that a large number of potential customers will reach his website, and hopefully decide to buy something.

Thus, if an advertiser could choose between two different keywords he would surely choose the one bringing him the more clicks while costing him the least. The rationale behind this basic idea is that some words have a large number of interested agents; this tends to increase the cost of each click. From the viewpoint of a "new" advertiser, he would surely be interested in looking for similar terms that have less competition. The search engines already make some suggestions on alternative keywords, or enable the advertiser to bid only for queries arriving from specific locations.

Nonetheless the advertisers would probably be interested in using a mechanism that would suggest them a set of alternative keywords. To formalize this concept we can think of an agent that wishes to change a certain word w. Let's call α_w the estimated number of clicks on w (as given by the Traffic Estimator, for instance), and cpc_w the estimated cost for each click. Let v_w be the agent's valuation for w. The average cost of bidding on this keyword is thus

$$\mathrm{cost}_w = \alpha_w \mathrm{cpc}_w,$$

and the agent's average revenue on this term is

$$u_w = \alpha_w (v - \mathrm{cpc}_w).$$

The objective of the agent is to find a subset of related words (that we will generically call synonyms), that have a higher estimated value while costing him no more than the original word. This binary knapsack problem can be written as

$$\begin{aligned} \max_S \quad & \sum_{i \in S} \alpha_i v_i \\ \text{s.t.} \quad & \sum_{i \in S} \mathrm{cost}_i \leq \mathrm{cost}_w \qquad \qquad (\mathrm{K}) \\ & S \subseteq \{\mathrm{synonyms}_w\}. \end{aligned}$$

If the valuation of the subset S is greater than the valuation of the original word $\alpha_w c_w$,

$$\sum_{i \in S} \alpha_i v_i \geq \alpha_w v_w,$$

S is the desired subset, since then

$$\sum_{i \in S} \alpha_i v_i - \sum_{i\ inS} \alpha_i \mathrm{cpc}_i \geq \alpha_w v_w - \alpha_w \mathrm{cpc}_w,$$

or equivalently $u_{i \in S} = \sum_{i \in S} \alpha_i (v_i - \mathrm{cpc}_i) > u_w$.

6.1.1 Synonyms

One of the aims of the simulator was to investigate the behavior of ad auction mechanism in the presence of agents who make use of keyword spreading. To model such behaviors we need a set of synonyms for each word.

We explore two alternative ways of performing keyword spreading. One uses the well known Wordnet ontology, the second is based on clustering web pages related to a query as found by a generalist search engine (in our case Google). The two resulting word distributions are different but the measured trends are consistent for both data sets, thus giving high confidence in the robustness of the experimental benchmark.

Wordnet The most important project for ontologies of words is *WordNet* (see Miller et al. [1990]). Originally proposed by the Cognitive Science Laboratory at Princeton University only for the English language, WordNet has become a reference for the whole information retrieval community, and similar projects are now available in many other languages. WordNet is a handmade semantic lexicon that groups words into sets of synonyms called *synsets*. Intuitively one can replace a word in a text with another from the same synset without changing its semantics. A word can appear in more than one synset if it has more than one meaning. Synsets are arranged as nodes in a graph such that there is an edge to connect two nodes if there is a relation between the two synsets. There are different types of possible relations, an exhaustive list of them can be found in the WordNet web site.[2] Given two synsets X and Y, the most common types of relations in WordNet are: *hypernym* if every X is a "kind of" Y, *hyponym* if Y is a "kind of" X, *holonym* if X is a part of Y and *meronym* if Y is a part of X. In our experiments we took into account only hypernym.

Clustering Google Data Given a query word, our goal is to find a set of semantically related words whose cost is lower than those of the query. We are not only interested to paradigmatic similarity, i.e., when two words may be mutually exchanged without effects on the semantics of the text, but also to syntagmatic similarity, i.e., when two words significantly co-occur in the same context. To achieve this goal we approach the problem as a *word clustering* (see Dhillon et al. [2002] and Li and Abe [1998]) task. Given a set of objects, clustering attempts to create a partition such that the objects in a cluster are related among them, while objects in different clusters are unrelated. Word clustering requires a corpus of documents related to the query word. To set up such a corpus we redirect the query to *Google* and download pages related to the first 100 results. Each page is later parsed and split to extract a set of sentences. Under the well established hypothesis that co-related words are more likely to stand in the same sentence, all the sentences not containing the query are discarded. We remove from each sentence over-represented words (stop words) that are often "syntactic sugar" and their removal does not affect the semantic content of the sentence. We added to the standard stop word list, a set of words that normally can not be considered stop words, but in the Web environment are considered generic (e.g. "download"). Once filtered, all the sentences are arranged in a term-document matrix whose rows correspond to sentences and whose columns to terms of the corpus. We tested different weighting schemes for terms, and we found that for our purpose a simple binary weighting scheme suffice. For clustering we

[2] `http://wordnet.princeton.edu/`

employed a fast implementation of the FPF clustering algorithm Gonzalez [1985]. As distance between pairs of words, i.e., columns of the term-document matrix, we used the well known cosine similarity. FPF is an iterative algorithm. It makes a new cluster at each iteration and populates it by extracting from the other clusters all the elements that are more related to the new cluster. The procedure stops when a given number k of clusters is reached. In our case it is impossible to predict a good value for k. Thus, instead of feeding k in advance, we make FPF check at each iteration the number of elements in the cluster containing the query word. When this number gets below a certain threshold (10 in our case) the algorithm stops and returns the list of the words in the cluster containing the query. This procedure ensures that we find a coherent cluster of words even if the query is not central in that cluster.

Table 6.1 gives some basic figures on the two resulting data sets, while Figure 6.1a and Figure 6.1b show the distribution of the number of synonyms per word and the distribution of the number of terms a word is synonym of.

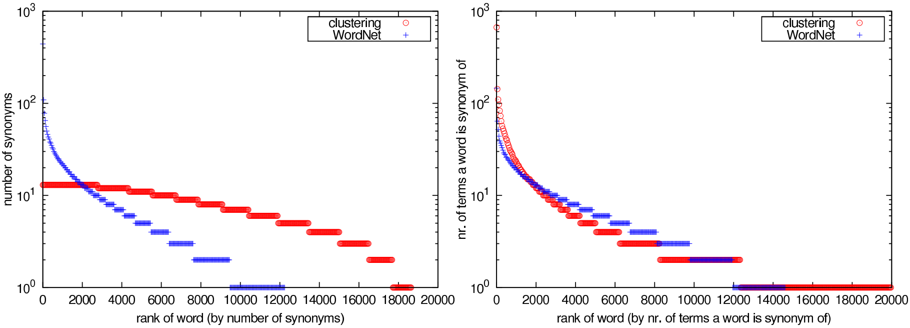
	clustering	Wordnet
Nr. of words with synonyms	18660	12271
Max. nr. of synonyms for a word	13	441
Max. nr. of terms a word is synonym of	668	146

Table 6.1: Some figures from the synonyms databases used.

There is a big difference in the boundary values of the databases: for example, there is a term for which Wordnet gives 441 synonyms. On the other hand, due to limitations in the computational resources, the clustering imposed a hard limit of 13 on the maximum number of synonyms per word. Nonetheless, as shown clearly by Figure 6.1a, the majority of the words have more synonyms in the clustering database than in the Wordnet one. Overall we can consider the databases comparable for our purposes.

6.1.2 Keyword Spreading

The simulator implements a basic version of keyword spreading. It allows certain agents, given a keyword w, to search for a subset of its synonyms that solves problem (K) (page 92) and has a greater value for the agent.



(a) number of synonyms per word

(b) number of words a term is synonym of

Figure 6.1: Comparison of the two synonyms databases used (in log-scale).

In the real world scenario, assuming each agent knows his private valuation for w v_w, the other parameters can be considered as given (for example by a tool like Google's Traffic Estimator). In our simulation we cannot directly use Google's data, since we already used it to initialize the word list. So, to have an estimate of the price of a keyword, the simulator maintains a slot-independent average of the cost of a click for each word: our estimated average CPC. This data can then be assumed to be an analogous to the one supplied by the Traffic Estimator.

The spreading starts only after a certain amount of queries has been performed (40% in the default settings). This happens to allow the simulator's estimated CPC for each keyword to stabilize.

When the simulation requires keyword spreading, one parameter determines the percentage of agents that will be allowed to apply this technique. When a query is made, it will be selected for keyword spreading only if

i) the prescribed amount of plain queries has already been made

ii) the current query has, among its interested agents, at least one that is allowed to swap words.

The simulator actually applies keyword spreading, only on a certain percentage of the queries that satisfy both these conditions. This is because we consider it as a rare event. When all these criteria are met, the simulator selects, uniformly at random, a single agent, among all the word's keyword spreading agents, and allows him to swap out the current query for a subset of its synonyms.

A further parameter that can be specified, is the percentage of words an agent is allowed to apply keyword spreading to. This is to rule out cases in which an agent would change all of his keywords, or would try to change back a synonym he had previously chose. The default value here is 8%. At this point, the query and the agent are considered valid, and the agent finally looks for a "good" subset of synonyms (if it exists).

Figure 6.2 compares the difference of the above simulation (whose logs are plotted in Figure 5.8) with an identical one in which we allow 20% of the agents to apply keyword spreading (using both the Wordnet database and our clustering techniques).

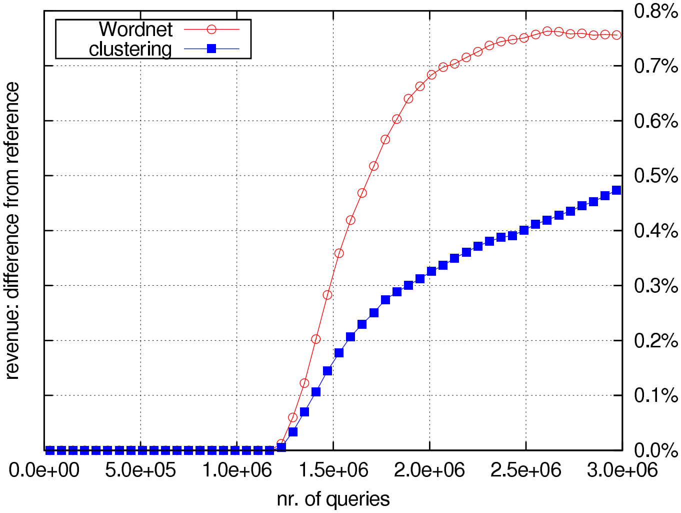


Figure 6.2: Comparison of the search engine's revenue when we allow some agents to change their keywords with synonyms provided by Wordnet database or by our co-clustering.

In figure 6.3, instead, we see the effect of keyword spreading for the agents that are allowed to change words, while figure 6.4 compares the revenue per query of agents that are not allowed to change words in the different scenarios.

The simulations point out that the use of keyword spreading is beneficial for all the involved parties: the search engine, the agents that apply the keyword spreading and also the agents who do not. Nonetheless

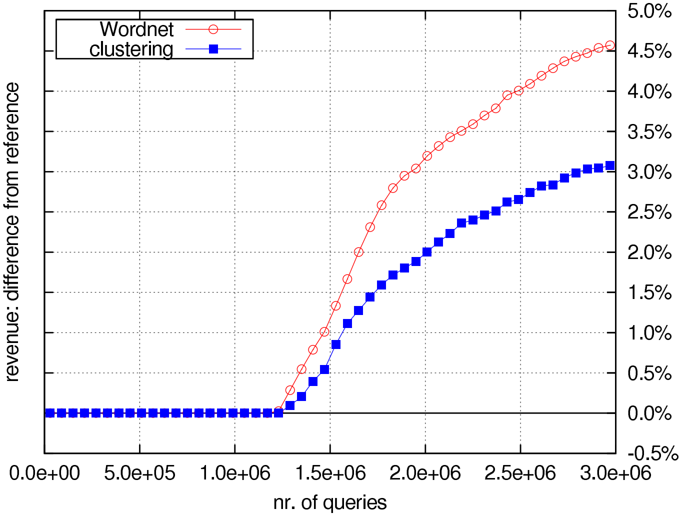


Figure 6.3: Comparison of revenue per query for agents that can change words, using both the Wordnet and co-clustering synonyms databases.

the agents that change some keywords are also the ones who increase their profit the most.

Failures

When an agent is given the possibility to change one of his keywords for similar words, not always is he able to find a set of synonyms that makes him better-off. Actually only a small percentage of the possible changes are successful (meaning that the agent leaves the original keyword for a set of its synonyms). When a keyword spreading substitution fails, two can be the causes:

i) the key has no synonyms,

ii) there is no subset of synonyms that satisfies the required properties.

The occurrence of case i) can be explained by looking at Table 6.1, since not all of the words have synonyms. The simulator records these occurrences, and for the above simulations they are reported in Table 6.2.

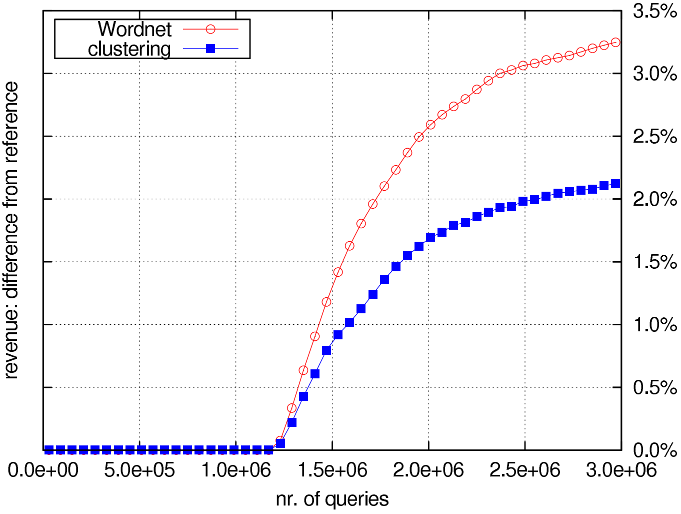


Figure 6.4: Comparison of revenue per query for agents that cannot change words, using both the Wordnet and co-clustering synonyms databases.

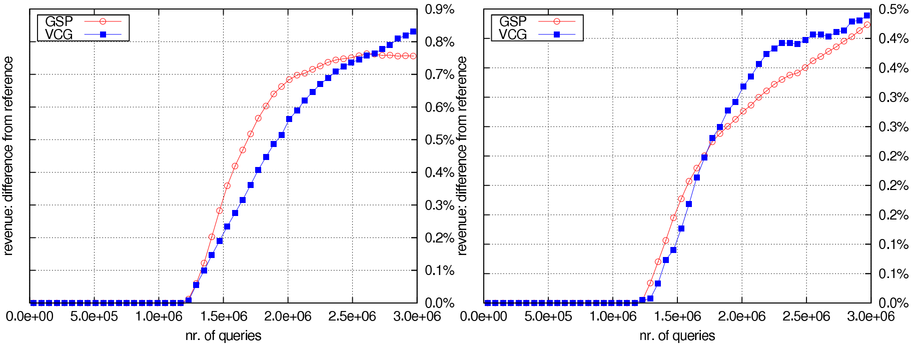
6.1.3 GSP and VCG

The above simulations were all run considering the GSP mechanism, we now would like to compare it to the VCG (as done in Figure 5.8).

There is basically no difference in the search engine's revenue under the two mechanisms (Figure 6.5). On the other hand, the GSP greatly increases the revenue for the changing agents (Figure 6.6), and also for the "normal" agents, although to a lesser extent (Figure 6.7).

	clustering	Wordnet
Nr. of tried word changes	208495	214201
Nr. of unique agents that changed words	42580	50473
Nr. of failed changes (% of total)	79.1%	74.7%
Changes failed due to lack of synonyms (% of failed)	51.4%	74.9%
Changes failed due to lack of good subsets (% of failed)	48.6%	25.1%

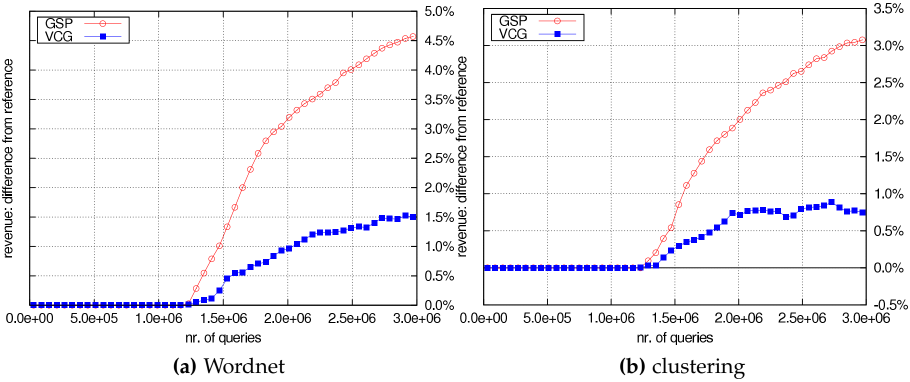
Table 6.2: Details on keyword spreading.



(a) Wordnet

(b) clustering

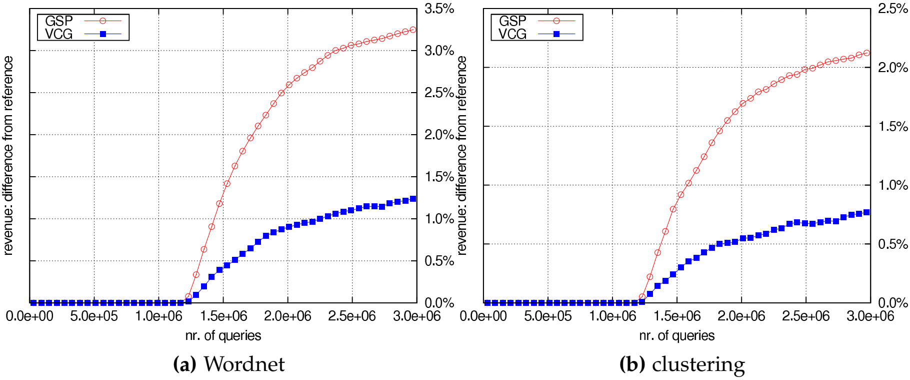
Figure 6.5: Google's revenue under the VCG and GSP mechanism, when 20% of the agents engage in keyword spreading.



(a) Wordnet

(b) clustering

Figure 6.6: Changing agents' revenue under the VCG and GSP mechanism, when 20% of the agents engage in keyword spreading.



(a) Wordnet

(b) clustering

Figure 6.7: Normal agents' revenue under the VCG and GSP mechanism, when 20% of the agents engage in keyword spreading.

6.1.4 Changing the Number of Strategic Agents

Another interesting analysis is to compare how the ad auction mechanisms react as the percentage of agents that are allowed to apply keyword spreading changes.

In the following simulations we gradually increase the percentage of agents that are allowed to use keyword spreading and see how this affects the results. Having fixed the percentage of agents, we perform 3 separate runs: one is the reference run,[3] and the other two are the runs performing keyword spreading, using the Wordnet or clustering databases.

Figures 6.8, 6.9, 6.10 show the changes in revenues for the search engine, the "normal" agents and the changing agents respectively.

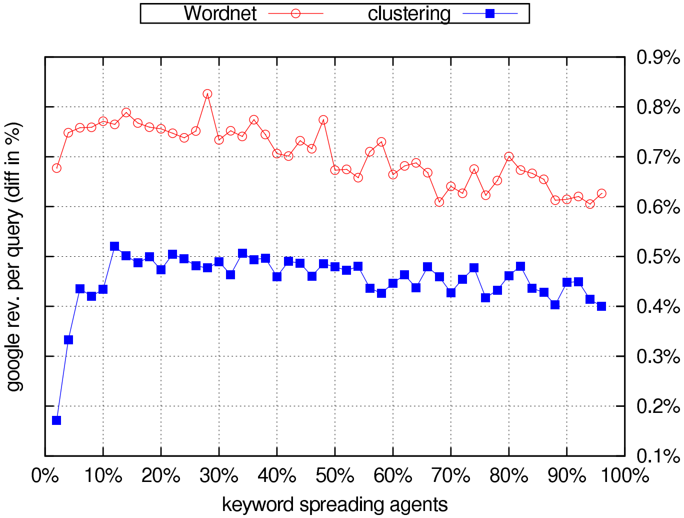


Figure 6.8: Search engine's revenue as the fraction of agents that engage in keyword spreading increases, using both the Wordnet and co-clustering synonyms databases.

Figures 6.12a, 6.12b show the percentage of fails due to either reason, while 6.11 shows the percentage of successful changes. All of the above figures refer to simulations performed using the GSP mechanism. The results obtained using the VCG mechanism have the same global

[3] The reference run is always performed; in this run agents are marked "changing", but are not allowed actually to perform keyword spreading. In this way, we can analyze separately the revenue of different kinds of agents.

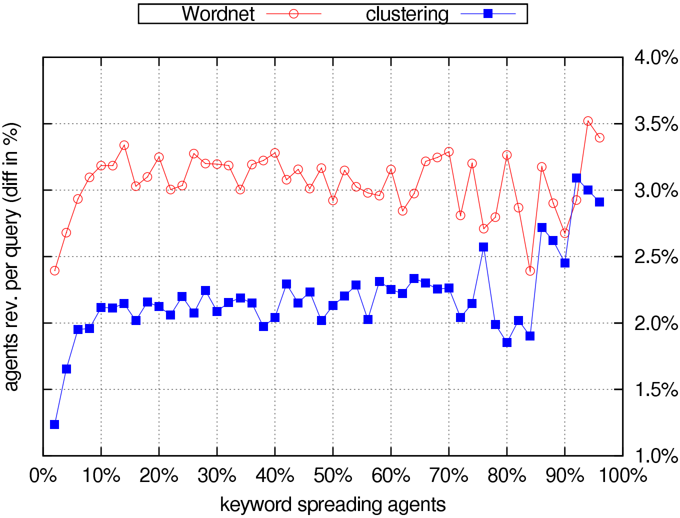


Figure 6.9: Agents' revenue for "normal" agents, as the fraction of agents that engage in keyword spreading increases, using both the Wordnet and co-clustering synonyms databases.

properties, although the differences from the reference run are usually smaller.

We can consider the revenue for the search engine and for the "normal" agents basically constant (Figures 6.8 and 6.9), while the keyword spreading agents' revenue is greatly affected by the percentage of "competition" (see Figure 6.10). An interesting detail is that the number of successful changes decreases slightly (Figure 6.11), as the number of failures to find synonyms for a key increases (Figure 6.12a). This is probably due to the fact that more words are looked up as the number of agents that can change words increases, and this highlights the fact that both databases have synonyms just for a part of the entire wordlist.

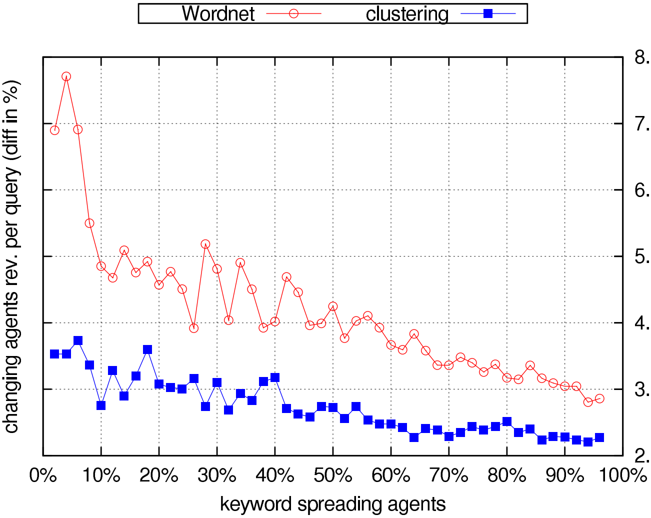


Figure 6.10: Agents' revenue for keyword-spreading agents, as the fraction of agents that engage in keyword spreading increases, using both the Wordnet and co-clustering synonyms databases.

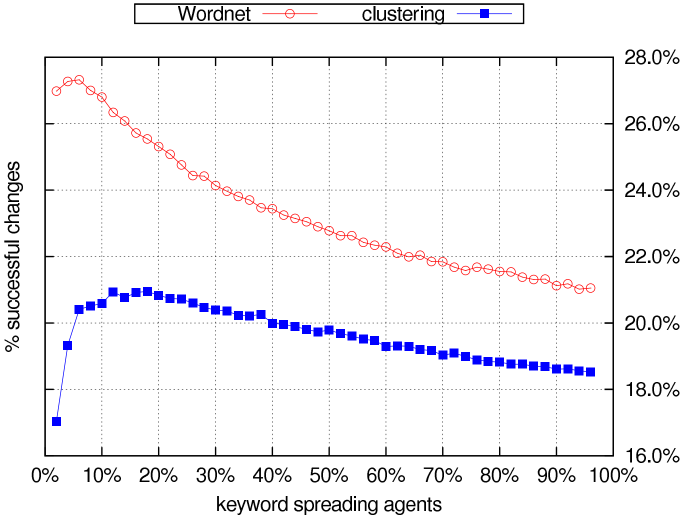
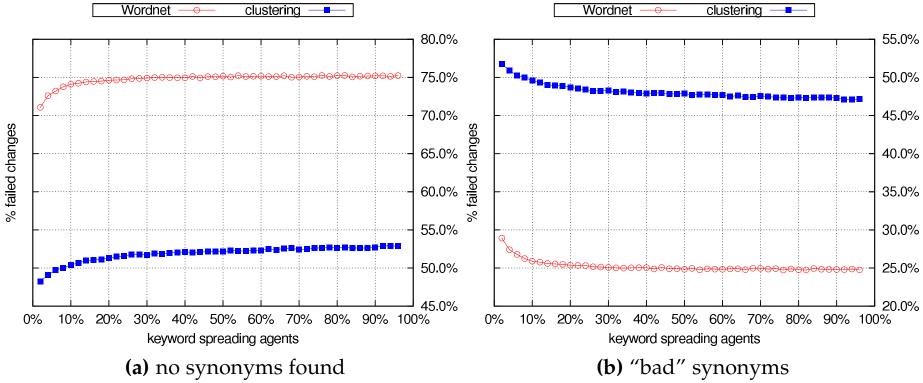


Figure 6.11: Percentage of successful changes, as the fraction of agents that engage in keyword spreading increases, using both the Wordnet and co-clustering synonyms databases.



(a) no synonyms found

(b) "bad" synonyms

Figure 6.12: Percentage of failures, i.e. agents that did not find a suitable set of synonyms, as the fraction of agents that engage in keyword spreading increases, using both the Wordnet and co-clustering synonyms databases.

6.2 Starvation Strategies

The simulator also implements other strategic behaviors. Specifically it implements the "starvation" strategy, described in Section 4.2 [page 74].

This strategy is implemented on a per-word basis: a certain percentage of keywords is assumed to have one (or more) strategic agents. This approach is in contrast with the one chosen for the keyword spreading, in which we select a percentage of agents to have the keyword spreading ability. The rationale behind this is that the strategic effect of agents is meaningful only if they are visible (i.e. appear in the slots). On the other hand, keyword spreading is beneficial even for agents that do not appear in the slots: they may swap that word out, and bid on different ones and still gain an advantage. So, to avoid having "useless" strategic agents, we chose a per-word implementation.

When a word is selected as having strategic agents, the last visible bidder is chosen as a strategist. In this way we maximize the effect of strategies, since he will be able to increase his bid and rapidly consume his opponent's[4] budget. The fact that we choose the last visible agent as the strategic agent for the word corresponds to a worst case scenario, since this is the agent that can potentially gain the most from starving his opponents.

The main objective of a strategic agent in our scenario is to bid just below the agent above him: in this way the attacked agent will spend his budget quicker, since each click will cost him more. There are two approaches in implementing this behavior. A "realistic" one, in which the agent increases his bid based on his actual position: until he is below his opponent he increases the bid, and as soon as he appears above his opponent he decreases it slightly. On the other hand, we can assume that the attacking agents simply knows the bid of the attacked agent, and can thus immediately set his bid just ε away from that. Clearly this second case is unrealistic, but again the idea behind it is to have a worst case scenario.

The first experiments (using the default data-set), compare the results of a normal run, with one in which 20% of the words have one (and only one) strategic agent. We use both the realistic and the unfair strategy implementations.

Figure 6.13 shows the percentage change in the search engine's revenue. Figure 6.14a shows the percentage change in the agents' revenue, only for the agents that actually apply strategies, while Figure 6.14b

[4] The bidder appearing immediately above him.

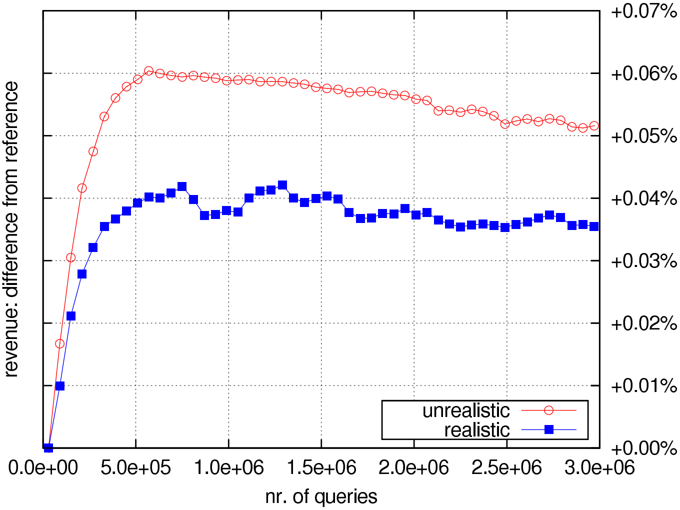
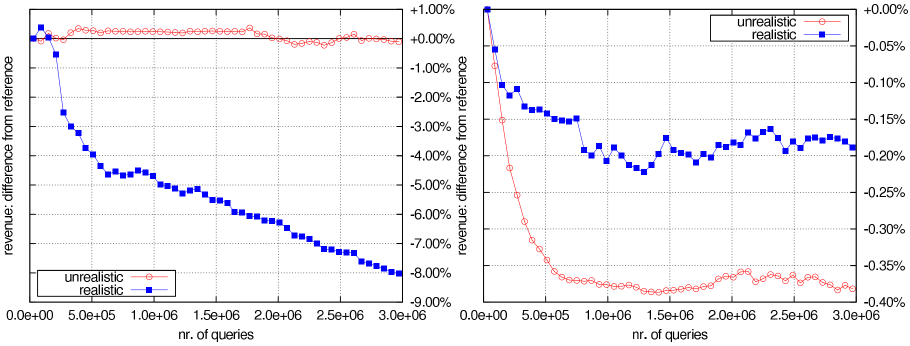


Figure 6.13: Difference in search engine's revenue, when 20% of the agents engage in the starvation strategy, using both the realistic and unrealistic bid discovery behaviors.

regards the revenue of non-strategic agents.

While the search engine's revenue is always (albeit slightly) increased, since the payments are always higher, it is interesting to see that, even using the unrealistic behaviors, strategic agents basically have no benefit (Figure 6.14a). Furthermore, if we consider the realistic behavior, the strategic agents are actually harmed. This happens mainly due to the fact that the clicks they might receive when they are above their opponent (trying to empirically determine his bid), have a negative revenue for them. The experiments show that this negative effect is greater then the benefit they obtain once their opponent is out of the competition. On the other hand, if we consider non-strategic agents (Figure 6.14b), we can see that they are surely harmed by the introduction of strategic agents, especially if these use the unrealistic behavior, however their loss is always under 1%.

Probably the effect of strategies could be emphasized by doing more queries, or increasing the number of strategic agents per-word. However one has also to consider the mild role of budgets in ad auctions. In fact agents can change their budget during the day, and so a strategic agent is not assured that he will eventually win the slot above him. This further decreases the effectiveness of this kind of strategies.



(a) strategic agents

(b) non-strategic agents

Figure 6.14: Difference in agents' revenue, when 20% of the agents engage in the starvation strategy, using both the realistic and unrealistic bid discovery behaviors.

Click Fraud Another phenomenon, based on the starvation principle we just described, is called **click fraud**.[5] In this case a bidder, instead of trying to raise his competitors' prices by increasing his bid, will simply generate a lot of clicks on their ads. This is apparently a much smarter strategy, since the attacker can directly control the cost of the attack, as opposed to the previous strategy in which it might happen that the attacker incurs in greater costs due to overbidding his opponent.

However, probably the most widespread use of click fraud has been in another context. In other advertising programs ads appear on external web pages, and not only the search engine's results page.[6] In this case, the owner of the external web page, will get a part of the revenue every time an ad gets clicked. In this setting, it would be in the web page owner's interest to click on ads on his page: each click costs him nothing (besides the time spent clicking), and generates revenue.

We did not investigate these kind of strategic behaviors since, in our experimental framework, they would have uninteresting consequences. The main research area in this topic is finding and implementing methods that can effectively tell a fraudulent click from a real one.

[5] `http://goo.gl/AyycI`

[6] See, for instance, the Google AdSense program `http://goo.gl/gHBVA`

6.3 Click Through Rate Estimation[7]

The click through rate is probably the most fundamental parameter in sponsored search, both from a practical point of view as well as from a theoretical standpoint.

From the point of view of an advertiser, the click through rates determine both the cost and the effectiveness of his campaign. Furthermore, in the theoretical auction model (see Chapter 2 [page 27]) it is assumed that click through rates are common knowledge, i.e. not only that every advertiser knows them, but also that he knows that everybody else knows them, and that everybody knows that everybody knows them, and so on. Most of the literature points out that, given enough time and money, agents would be able to learn click through rates. For instance, an agent might change his bid in order to appear in each slot, and for every slot gather enough information as to get an estimate for the corresponding click through rate. However, as the results in 6.2 suggest, this might be very costly for an agent, since, assuming we start from an equilibrium condition, the cost of the clicks he receives when investigating the click through rate for slots above him, would exceed his valuation. This would be amplified by the fact that, to get reliable values, an advertiser would have to receive quite a number of clicks and average them.

Considering our simulator, the way in which click through rates were used, was probably its weakest point. We thus developed a simple and reliable external algorithm that allows to roughly estimate click through rates for a given keyword. We successively ran this algorithm for all the keywords in our dataset and included these estimated click through rates in the simulations.

6.3.1 Previous work: CTR Estimation

The Click Through Rate (CTR) can be defined for any type of displayed search result (such as news, non-sponsored search results, and sponsored search results) and several methodologies have been proposed for the estimation problem on these data. Here we concentrate just on CTR for sponsored search results (aka ads) since this problem has somewhat special requirements not shared by the other types of displayed items. More generally, one can view CTR as just one particular indicator a SE can extract form user-logs and view CTR estimation as a special case

[7] The work presented in this section has already appeared in publication, see Budinich et al. [2011].

of a more general problem of modeling advertiser-specific long term user-behavior Archak et al. [2010].

The Click Through Rate (CTR) of a sponsored advertisement is one of the key quantities that influence revenue for the search engine, thus, in selecting the ads to be displayed among those bidding for a keyword, considerations based on CTR are paramount. For ads with a long time series of display/click events the search engine can use frequencies to approximate probabilities. However even a search engine faces a challenge when trying to evaluate/approximate a rare ad, or a new ad, or an ad for which new keywords are associated. In these cases historical data is missing or too scant, or unreliable. Regleson and Fain Regelson and Fain [2006] propose a method based on the idea of clustering both rare and frequent ads by similarity of the set of associated bidding keywords so that the CTR of rare ads can be inferred from the CTR of the frequent ads in the same cluster. Richardson et al. Richardson et al. [2007] use a logistic regression model based on a collection of 1M ads as well as many associated search engine measurements taken form the Microsoft search engine. A monotone regression methodology is proposed in Gluhovsky [2010].

Actually, the CTR is influenced by two rather different factors. The first factor is the display position of the sponsored search result on the screen, as it is often assumed that a display in top slots leads to higher CTR (caeteris paribus). Joachims et al. Joachims et al. [2007] describe a user-study with eye-tracking devices so to correlate user attention and user actions and model the effect of positional bias in user preferences. Several models of positional bias in user click-logs are compared in Craswell et al. [2008]. A Markovian model for predicting click-position correlation is proposed in Aggarwal et al. [2008]. In our analysis we do not model position, and we deal instead with the second factor that is the intrinsic relevance of the ad to the presumed specific need of the user as expressed in the query. It is a common assumption that relevant high quality ads get more clicks than irrelevant or low quality ads[8].

Among the several methods that address the ad relevance factor we mention Agarwal et al. [2007], Dembczynski et al. [2008], Ciaramita et al. [2008], Shaparenko et al. [2009], Hillard et al. [2010]. All these methods for CTR estimation and a few the others we encounter in literature make use of search engine historical data in direct or indirect form and of measurements that only a search engine can make, thus we can collectively term these methods as “internal". In the spirit of Impres-

[8] For sake of precision we should note that in some cases this is not true, since the bidder may aim for non-specific, general needs of human beings, such as sex, love, loosing weight, etc.. In our analysis we neglect these “non-specific” advertising strategies.

sionrank Bar-Yossef and Gurevich [2009] we estimate the CTR by using an “external" approach, that uses only information routinely made public by the search engine. This is important since CTR estimation is a key parameter also for other key players, besides search engine, such as advertisers that need CTR estimates to optimize their ad campaign strategies. At the best of our knowledge, our external approach to CTR estimation has not been proposed in literature so far.

Since both the ad textual representation and the query tend to be rather short, thus limiting the power of machine learning-based methods, we use a form of query augmentation Broder et al. [2008], Broder et al. [2009]: we use the search results returned by the search engine in response to a query as its semantic expansion in applying ML tools.

6.3.2 Learning CTRs

Our algorithm is driven by the intuitive idea that a user is more likely to click on an ad if she/he finds its content interesting according to the query. Following this intuition, for an ad, the higher its correlation with the query, the higher its click through rate. A simple way to evaluate the correlation between the ad and the query is to measure the degree of membership of the ad to the class defined by the snippets returned by the search engine as result of the query. The main advantage of this approach is that the correlation between the query and the search results is guaranteed by search engines. A key point is to select a classifier: it must be reliable even with few (in the order of a hundred) positive training samples, its output should not be a binary value, but a real number in a bounded domain, which we can interpret as a probability. The second point is essential, since if we were to use a classifier with binary output we would have no way of distinguishing among ads with the same classification. We chose a Bayesian classifier. The outcome of this kind of classifier is a real number in the range $[0, 1]$ that could be interpreted as the probability of the ad to belong to the class (i.e being relevant according to the query).

A key point in our approach is that we use a different classifier for each query. This is because we are not interested in trying to estimate the best page for an ad, but only how well the ad fits in that particular page. Each of these classifiers is trained using as examples the textual snippets of the search results. To increase the number of examples we constrained the search engine to return the maximum number of results per page (namely 100 snippets), moreover to improve the significance of each training sample, stop words, symbols and numbers are removed. After the training is completed, each ad appearing on that page is given as an

Algorithm 1

```
procedure EstimateCTR($dict$)
    for each keyword $q \in dict$ do
        B ← new Bayesian classifier
        for every search result $r$ for $q$ do
            B.train($r$)                    ▷ Only $r$'s text snippet is used
        end for
        for every ad $a$ of $q$ do
            CTR[$q$][position($a$)] = B.predict(a)
        end for
    end for
    return CTR[$q$] $\forall q$
end procedure
```

input to the classifier which returns its relevance to the query. Again we consider only the textual fields of the ad, with a similar preprocessing as for the training samples. Once the probabilities have been estimated for all ads on the page, we normalize them to sum to one and store the profile along the query. The pseudo code in Algorithm 1 gives an outline of how the algorithm works.

6.3.3 Extending the Word List

As described in Section 6.3.2, our algorithm's input is a search engine's results page. For this reason, the first step in getting click through rateestimates for all our keywords was to download the results pages for all the entries in our wordlist (see Section 5.1.1 [page 78]). Interestingly Google displayed side ads for less than 9K of them. One possible reason for this low percentage of pages carrying ads is that the SCOWL word list contained only one term queries. Given the generality of such queries, it is likely that very few advertisers bid on them, also considering the fact that the search engine strongly encourages the users to choose more specific keywords (the experimental results on keyword spreading in Section 6.1 quantified why this might be a good choice for both the search engine and advertisers).

To increase the size of our dataset, we considered expanding our dictionary to multi word queries. However, any reasonable idea on how to combine the words in our original word list in multi term queries results in a combinatorial blow up. This is a problem, since search engines impose strict limits on the number of queries that can be made from a given IP address (or range of IP addresses).

To overcome this issue we used, once again, data available from the search engine itself. In particular, for each of the 8K terms that we knew displayed ads, we used Google's Autocomplete feature to get 4 more suggestions, which we hoped would contain ads themselves, being intuitively "popular" searches. After having downloaded the search results for these multi term queries and having merged it with the search results for the SCOWL list, we ended up with a word list of 19K results pages that displayed ads (see Table 6.3).

Wordlist	Entries	With ads
SCOWL	35867	8282 (23%)
Google's Autocomplete	31325	10947 (35%)

Table 6.3: The word list data.

A characteristic of sponsored search is that the number of slots and the ads displayed change over time. To quantify this observation we measured, during the course of roughly two weeks, the number of slots displayed by Google for three fixed queries (namely "loans", "lawyers" and "flights"), by repeating them every two hours. The results are shown in Figure 6.15. Very often also the ads themselves change over time. To mitigate the effect of these facts on our simulations, we downloaded another copy of the same dataset. For each keyword we then set the number of slots to be the minimum of the two observed values. We chose the minimum, instead of, say, the average, since this allows us to always have the final click through rate of a slot be an average of two values. To lighten the exposition, from now on, when we refer to the number of slots or CTR in the dataset, we actually mean the value obtained by combining the two datasets.

Next we collected specific data on the 19K keywords, using the Traffic Estimator website[9], as described in Section 5.1.2 [page 78]. The dataset's main characteristics are described in Table 6.4. A first analysis of the dataset, shown in Figure 6.16, shows the distribution of the number of slots per page (the expected cost per click and estimated search volume are very similar to the graphs for the original dataset, shown in Figure 5.2 [page 80]).

Finally we used our algorithm to estimate the CTRs of all the 19K search results pages downloaded on both datasets. We then averaged

[9] http://goo.gl/G5e1

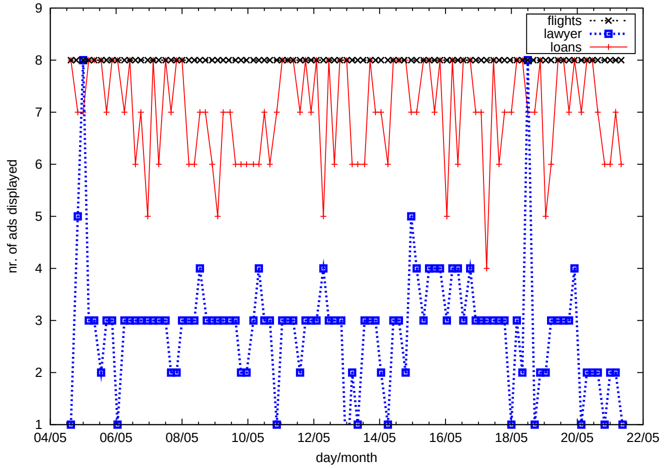


Figure 6.15: The number of slots for 3 queries over a period of roughly two weeks in spring 2009.

Number of entries	19229
Search Volume range	[151×10^6 - 1]
Cost per click range	[$139.31 - $0.01]
Number of slots range	[1-8]

Table 6.4: The dataset used.

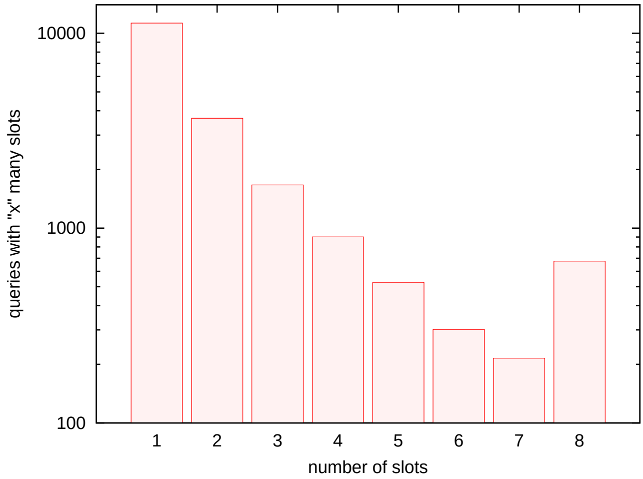


Figure 6.16: Histogram representing the number of slots on each of the 19K pages in the dataset (in log scale).

the values, obtaining 19K different CTR profiles. Since, during our simulations, we are only interested in queries that give rise to a click, we normalized the click through rate values so that the sum of CTRs on any page is 1. Figure 6.17 shows the distribution of average click through rates as the number of slots on the page changes.

Correlation As already discussed in Section 5.2.1 [page 80], we measured the Kendall τ rank correlation coefficient for the number of slots dataset against the estimated volume and estimated cost per click dataset.

Search volume vs. slots	0.1244
CPC vs. slots	0.1493

Again these values show that the correlation between these quantities is low: it is not generally true that words with high search volume (or cost) have more slots.

6.3.4 Including CTRs in the Simulator

The simulator used up to this point uses a fixed number of slots and a fixed CTR for each slot. We now describe how we included in the

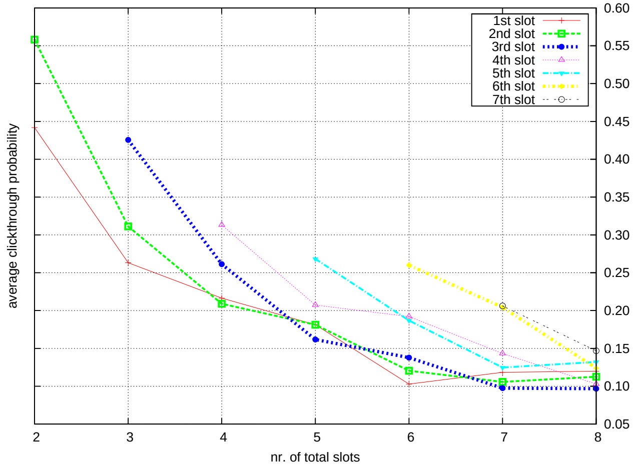


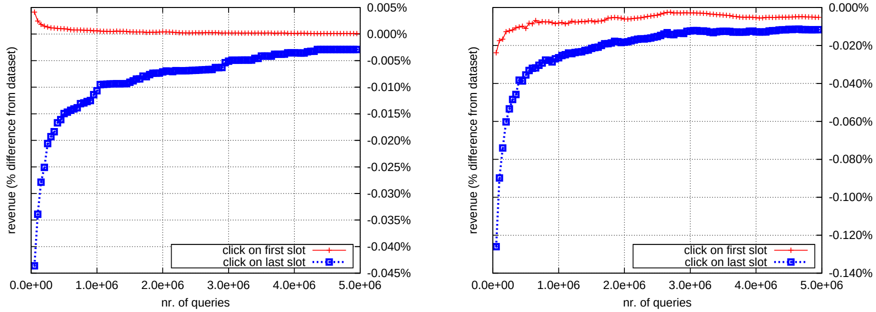
Figure 6.17: Click through rate for the single slots, as the number of total slots increases.

simulation the new data on click through rates and number of slots, and we present the outcomes of the related experiments.

The number of slots is used directly; when the simulator extracts a query, it looks it up in the database and displays exactly the same number of slots as in the results page that was downloaded.

When considering click through rates estimates, there are at least a couple of ways in which they might be used. A straightforward approach would be to use, for each keyword, the estimated click through rates found in the dataset. However, we believe, this is of limited interest. In fact the quality and type of ads displayed for any fixed query varies widely. For instance it often happens that entering the same query twice consecutively gives results pages with different ads. Furthermore, the number of slots itself changes over time (see Figure 6.15). Although, as described in Section 6.3.3, we already considered averaged CTR, this approach would yield results too dependent on the actual text of the ads that were displayed at the moment we downloaded our dataset.

To partially overcome this bias we implemented a different approach, in which we use CTRs to model agents' behavior. We first categorize all the results pages in our dataset by the number of slots they exhibit. We now have, for each number of slots j, a collection of m_j different profiles, where m_j is the number of pages in our dataset that have j slots (the values of m_j are shown in Figure 6.16 panel c). We now interpret



(a) Google's Revenue

(b) Agents' Revenue

Figure 6.18: Google's and agents' revenue change (in percentage) comparing extreme profiles in which users always click on the first/last slot to the actual profiles in the dataset.

this data as different profiles characterizing *agent* behavior. In particular, when the simulator picks a query q, it first checks how many slots query q has in the actual dataset, say j, and then it picks a random profile among the m_j ones available, and generates a click according to that probability distribution.

6.3.5 Experimental Results

Without access to search engine query logs, it is difficult to evaluate the quality of our CTRs. To give an idea of the changes in revenue when using different profiles we compare the profiles we gathered from our dataset with two "extreme" profiles: one in which users always click on the first slot, and one in which agents always click on the last slot.

Notice that when there is only one slot, there is no difference between any of the above profiles (given that our algorithm normalizes the CTR), which will all produce a click on the only ad displayed. For this reason, in the current simulations, we consider only pages displaying two or more ads.

The graphs in Figure 6.18 show the percentage change in revenue for the search engine (panel a) and the agents (panel b) when using the two extreme profiles, compared to the same measures when using the dataset profiles.[10] It is interesting to notice that, in this simple but im-

[10] All simulations start with 2×10^6 agents, and run for 5×10^6 queries.

portant measure (i.e. revenue), there is little change between these extreme profiles. However, although the percentage change is small, it must be remembered that, given the huge number of queries performed each day, these are actually big differences in monetary terms.

Part IV

Bounded Rationality

Chapter 7

Revealed Preference[1]

In the classical microeconomic setting there is a set of goods X and each agent is described by a utility function u and a budget b. Given a set of prices p, the objective of each agent is to find an affordable bundle that maximizes his utility, i.e. to solve the following optimization problem

$$\begin{aligned} \max_{x \in \mathsf{X}} \quad & u(x) \\ & p \cdot x \leq b. \end{aligned} \tag{7.1}$$

We can now associate, to each price-budget couple (p, b) the set of bundles that solve (7.1). This rule is usually denoted as $x(p, b)$ and called **Walrasian demand correspondence**.

We will throughout assume that the preference relation $\succeq$ underlying u is **locally non-satiated**, i.e. that for every bundle x there is an arbitrarily close bundle y such that $y \succ x$. This property directly implies that, if x^* is a feasible and optimal solution to (7.1), then $p \cdot x^* = b$, i.e. the consumer spent his whole budget. To see why assume that is not the case: we can now use the unspent portion of the budget to buy a bundle that is better than our optimal solution (since by local non-satiation such a bundle must exist and we can pick it close enough to be affordable), contradicting the fact that it was an optimal solution.

Assume now that we are able to arbitrarily set m different price vectors $(p^1, p^2, \cdots, p^m)$. At each different price we will observe a different demand: call x^i the demand at prices i, i.e. $x^i = x(p^i, b)$. For simplicity we assume that the budget is the same at all m price points. The observed price-consumption points $\left\{ \left((p^i, x^i \right) \right\}_{i=1}^{m}$ satisfy the **generalized axiom**

[1] The work presented in this chapter is an ongoing research project, see Budinich et al. [2010b].

of revealed preference. The axiom states that, for any list $\left\{ \left((p^i, x^i \right) \right\}_{i=1}^n$ with the property that

$$p^j \cdot x^{j+1} \leq p^j \cdot x^j \quad \text{for all } j \leq n-1 \tag{7.2}$$

it must be the case that $p^n \cdot x^1 \geq p^n \cdot x^n$. Equation (7.2) can be interpreted as follows: since x^j was the optimal bundle at prices p^j, if $p^j \cdot x^{j+1} \leq p^j \cdot x^j$ then also x^{j+1} was an affordable bundle at prices x^j. This implies that the consumer, by choosing x^j, revealed that he prefers it to x^{j+1} (since otherwise he could have picked that): $u(x^j) \geq u(x^{j+1})$. By transitivity, a chain of these inequalities implies that $u(x^1) \geq u(x^n)$. Now if the condition required in the axiom weren't true, i.e. $p^n \cdot x^1 < p^n \cdot x^n$, this would imply that $u(x^n) > u(x^1)$, a contradiction. This informal argument shows that, any sequence of observations arising as solutions to (7.1) must satisfy this axiom.

If we form a graph G whose vertices are the observation points, the generalized axiom of revealed preference has a simple graph theoretic interpretation. Assign weight a_{ij} to edge $e = (i, j)$, where

$$a_{ij} = p^j \left(x^{j+1} - x^j \right).$$

Now (7.2) is simply $a_{ij} \leq 0$, and the generalized axiom of revealed preference can be restated as

> every negative length cycle in G has at least an edge with positive weight.

Up to now we have seen what necessary conditions must be satisfied by data arising from a utility maximizing consumer. However the inverse problem, of determining if a given data is consistent with a rational agent maximizing his utility, is also fundamental.

In 1967 Afriat (see Afriat [1967]) showed that the generalized axiom of revealed preference is also a sufficient condition for a set of observations to be the output of an agent solving problem (7.1). Furthermore the proof is constructive, exhibiting a monotone and concave function that **rationalizes** the data. We can state Afriat's theorem as follows.

Theorem 7.1 (Afriat's Theorem). *Given a set of observations $\left\{ \left((p^i, x^i \right) \right\}_{i=1}^m$ the following conditions are equivalent*

i) the data satisfies the generalized axiom of revealed preference,

ii) there exist positive values $\omega^1, \cdots, \omega^m$ and $\lambda_1, \cdots, \lambda_m$ such that

$$\omega^j \leq \omega^i + \lambda^i p^i (x^j - x^i), \tag{7.3}$$

iii) *there exist a non-satiated, concave, strictly monotone and continuous utility function that rationalizes the data.*

The equivalence between ii) and iii) is easy to see. Simply define

$$u(x) = \min\left\{\omega^1 + \lambda^1 p^1(x - x^1), \cdots, \omega^m + \lambda^m p^m(x - x^m)\right\}.$$

The function $u(\cdot)$ is a pointwise minimum of a collection of increasing linear functions, and is thus itself concave, strictly monotone and continuous. Furthermore it rationalizes the data, since

i) at x^j, $u(x^j) = \omega^j$ (by (7.3) and the definition of $u(\cdot)$),

ii) if $p^j \cdot x \leq p^j \cdot x^j$, then $u(x) \leq u(x^j)$. To see this notice that $u(x) \leq \omega^j + \lambda^j p^j(x - x^j) \leq \omega^j = u(x^j)$.

For the rest of the proof we refer to Afriat [1967], Varian [1982], Piaw and Vohra [2003], Fostel et al. [2004].

7.1 Non Rationalizable Observations

Data gathered from experiments and field based observations are sometimes inconsistent with the generalized axiom of revealed preference, and thus cannot be the output of an agent maximizing his utility. Assuming agents are instead maximizing their utility, this discrepancy between the theoretical results and the real world could be explained by hard-wired biases (such as those towards the *status quo*) or decision making errors.

However, if the relevant set of inequalities (i.e. (7.2)) is violated, existing theorems give no information on the nature of the underlying preferences. If the violations are small in an appropriate sense, it seems plausible that the underlying preferences should be approximately consistent with maximizing a concave utility function.

The first problem in this approach is that, in the classical models, an agent is described by his preference relation. It is unclear what an approximation to such a preference relation might be. Switching to utility functions is of little help. In fact, the function $u(\cdot)$ represents preferences $\succeq$ if $u(a) \geq u(b) \iff a \succeq b$. Now pick any monotone function f: $f(u(\cdot))$ still represents $\succeq$. To overcome this limitation we restrict the class of preferences to the ones that admit a linear utility function representation. In this case we can guarantee, if the coefficients are normalized, that, if a preference relation can be represented by such a function, the representation is unique.

The problem of dealing with non-rationalizable observations has already been considered. One approach (see Houtman and Maks [1985]) discards a set of observations so that the remaining ones satisfy the generalized axiom of revealed preference. Using this approach the main problem becomes determining which observations to discard. Afriat himself (see Afriat [1972]) had considered his problem, by defining a **critical cost index** that, by relaxing the budget constraints, would allow one to make the observations rationalizable. In the next section we describe our approach, and present some preliminary results that relate it to Afriat's proposed solution.

7.2 Approximate Rationalizability

Let G be a finite set of distinct goods and b the budget of the agent. Let $p \in \Re^{|G|}$ be a non-negative price vector and $x \in \Re^{|G|}$ be a non-negative demand vector. Suppose a collection of price-demand observations: $\{p^j, x^j\}_{j=1}^n$. We would like to know if there is a non-negative vector $v \in \Re^{|G|}$ such that

$$x^j \in \arg\max\{\sum_{i \in G} v_i x_i : \sum_{i \in G} p_i^j x_i \le b,\ 0 \le x_j \le 1\ \forall i \in G\}\ \forall j.$$

It is straightforward to see that the observations $\{p^j, x^j\}_{j=1}^n$ can be rationalized by a suitable collection of non-zero and non-negative v_j's if the following system has a feasible solution:

$$\frac{\sum_{i=1}^n v_i x_i^j}{b} \ge \frac{v_k}{p_k^j}\ \forall j,k \tag{7.4}$$

$$\sum_i v_i = 1 \tag{7.5}$$

$$v_i \ge 0\ \forall i \tag{7.6}$$

If the system (7.4-7.6) is infeasible, then the observations $\{p^j, x^j\}_{j=1}^n$ cannot be rationalized assuming that the agent is maximizing a linear utility function. What if the system is 'close to feasible'? Is there a sense in which the observations are the result of behavior that is approximately rational?

To explore this possibility we consider a relaxed version of the system (7.4-7.6) above:

$$\frac{\sum_{i=1}^n v_i x_i^j}{b} \ge r\frac{v_k}{p_k^j}\ \forall j,k \tag{7.7}$$

$$\sum_i v_i = 1 \tag{7.8}$$

$$v_i \geq 0 \ \forall i \tag{7.9}$$

Here $r \in [0,1]$ is a parameter that quantifies how much system (7.4-7.6) must be relaxed to ensure feasibility. Straightforward binary search will determine the largest value of r for which system (7.7-7.9) is feasible.

Fix a value of r for which system (7.7-7.9) is feasible and v^* any feasible solution.

Theorem 7.2. *Let* $y^j \in \arg\max\{\sum_{i \in G} v_i^* x_i : \sum_{i \in G} p_i^j x_i \leq b,\ 0 \leq x_i \leq 1\ \forall i \in G\}\ \forall j$. *Then,*

$$\frac{\sum_{i \in G} v_i^* x_i^j}{\sum_{i \in G} v_i^* y_i^j} \geq r.$$

Proof. Observe that

$$\sum_{i \in G} v_i^* y_i^j = b \max_{i \in G}(\frac{v_i^*}{p_i^j}) \leq br^{-1} \frac{\sum_{i \in G} v_i^* x_i^j}{b} = r^{-1} \sum_{i \in G} v_i^* x_i^j.$$

□

In words, the observations $\{p^j, x^j\}_{j=1}^n$ can be rationalized by assuming that the agent has selected an approximately optimal bundle at each price. Notice that we measure approximate optimality in a relative sense. Such a measure is sensitive to the choice of utility function. For example, if utilities were affine rather than just linear, one would make a nonsense of the ratio in Theorem 1.

Afriat [1972] also considered the question of how to handle a set of observations that is not rationalizable. He proposed a critical cost efficiency index that measures how much one must relax each budget constraint (in the set of observations) in order for the observations to be rationalizable. We imitate Varian's (see Varian [1994]) description of how the index is computed.

Choose a number $e \in [0,1]$ and set up a directed graph $\hat{D}(e)$ with one vertex for each observation. An edge is directed from observation i to j, denoted (i,j), if $ep^i \cdot x^i \geq p^i \cdot x^j$. In words, in observation i, when x^i was chosen, x^j was not only a feasible choice but cheaper by a factor of e. Thus, x^i is revealed to be directly preferred to x^j. Denote by $D(e)$ the transitive closure of $\hat{D}(e)$. $D(e)$ records both the direct and indirect preferences.

The graph $D(e)$ satisfies the generalized axiom of revealed preference (GARP) if for all directed edges (i,j) in $D(e)$, $p^j \cdot x^i \geq ep^j \cdot x^j$. The

Afriat critical cost index is the largest number e^* such that $D(e^*)$ satisfies GARP. It is interpreted to mean that at each observation we allow the agent to sub-optimize to the extent that they 'burn' a fraction $(1 - e^*)$ of their budget.

Theorem 7.3. *Let r^* be the largest number for which system (7.7-7.9) is feasible and v^* any feasible solution. Then, $e^* \geq r^*$.*

Proof. It suffices to show that $D(r^*)$ satisfies GARP. Let $1 \to 2 \to \ldots \to k$ be a directed path in $\hat{D}(r^*)$. Since $(j, j+1) \in \hat{D}(r^*)$ it follows that $r^* p^j \cdot x^j \geq p^j \cdot x^{j+1}$ for $j \leq k-1$. We claim that this implies that $v^* \cdot x^j \geq v^* x^{j+1}$. In particular, $v^* \cdot x^1 \geq v^* \cdot x^k$. Suppose not.

Let $y^j \in \arg\max\{v^* \cdot x : \text{ s.t. } p^j \cdot x \leq b\}$ for all j. Observe that

$$r^* y^j \in \arg\max\{v^* \cdot x : \text{ s.t. } p^j \cdot x \leq r^* b\}.$$

Since $r^* b \geq r^* p^j \cdot x^j \geq p^j \cdot x^{j+1}$, it follows that x^{j+1} is a feasible solution to the problem $\max\{v^* \cdot x : \text{ s.t. } p^j \cdot x \leq r^* b\}$. Hence,

$$v^* \cdot x^{j+1} \leq v^* \cdot r y^j \leq v^* \cdot x^j$$

which contradicts the assumption that $v^* \cdot x^j < v^* x^{j+1}$.

Since $1 \to 2 \to \ldots \to k$ is a directed path in $\hat{D}(r^*)$, the edge $(1, k)$ will be in $D(r^*)$. To prove that $D(r^*)$ satisfies GARP we must show that $p^k \cdot x^1 \geq r^* p^k \cdot x^k$. Suppose not, i.e., $p^k \cdot x^1 < r^* p^k \cdot x^k$. This means x^1 is a feasible solution to $\max\{v^* \cdot x : p^k \cdot x \leq r^* b\}$. Hence $v^* \cdot r^* y^k > v^* \cdot x^1$. Strict inequality follows from the supposition that $p^k \cdot x^1 < r^* p^k \cdot x^k \leq r^* b$.

However, $v^* \cdot x^k \geq r^* v^* \cdot y^k$. Therefore,

$$v^* \cdot x^k \geq v^* \cdot x^1 \geq r^* v^* \cdot y^k > v^* \cdot x^1 \geq v^* \cdot x^k$$

a contradiction. □

We give an example showing $r^* \neq e^*$. Specifically, $r^* < \frac{1}{k} e^*$ for any $k > 1$. Choose a value of r such that (7.7-7.9) is feasible. We derive some general bounds on r for the simple case consisting of just two observations: $\{p^{(1)}, x^{(1)}\}$ and $\{p^{(2)}, x^{(2)}\}$. Assuming they arise from an agent maximizing a linear utility, we can suppose they lie on the boundary of the budget set

$$x^{(1)} = (x_1^{(1)}, 0), \qquad x^{(2)} = (0, x_2^{(2)}).$$

We also assume that the agent's budget is the same in both observations, $p_1^{(1)} \cdot x_1^{(1)} = p_2^{(2)} \cdot x_2^{(2)} = b$. For this simple example we can list all inequalities in (7.7):

$$\begin{aligned}
\frac{vx_1^{(1)}}{b} &\geq r\tfrac{v}{p_1^{(1)}} \quad j=1, \quad k=1 \\
\frac{vx_1^{(1)}}{b} &\geq r\tfrac{1-v}{p_2^{(1)}} \quad j=1, \quad k=2 \\
\frac{(1-v)x_2^{(2)}}{b} &\geq r\tfrac{v}{p_1^{(2)}} \quad j=2, \quad k=1 \\
\frac{(1-v)x_2^{(2)}}{b} &\geq r\tfrac{1-v}{p_2^{(2)}} \quad j=2, \quad k=2.
\end{aligned}$$

Substituting for b, we see that the first and last inequalities are always satisfied, while the other two become

$$\frac{v}{1-v} \geq r\frac{p_1^{(1)}}{p_2^{(1)}} \tag{7.10}$$

$$\frac{v}{1-v} \leq \frac{1}{r}\frac{p_1^{(2)}}{p_2^{(2)}}. \tag{7.11}$$

Thus (7.7-7.9) will be feasible when

$$r^2 \leq \frac{p_1^{(2)}}{p_2^{(2)}}\frac{p_2^{(1)}}{p_1^{(1)}}. \tag{7.12}$$

Notice that, if we set $r = 1$ in (7.10-7.11) we obtain the conditions under which there is a solution to system (7.4-7.6), namely $\frac{p_1^{(2)}}{p_2^{(2)}}\frac{p_2^{(1)}}{p_1^{(1)}} \geq 1$.

It is now straightforward to find a set of observations for which $r \leq 1/k$. For example, consider

$$\begin{aligned}
p^{(1)} &= (k^2, k^2) \quad x^{(1)} = (1, 0) \\
p^{(2)} &= (1, k^2) \quad x^{(2)} = (0, 1),
\end{aligned}$$

where $k > 1$. The budget is $b = p^{(1)} \cdot x^{(1)} = p^{(2)} \cdot x^{(2)} = k^2$. Since $\frac{p_1^{(2)}}{p_2^{(2)}}\frac{p_1^{(1)}}{p_2^{(1)}} = \frac{1}{k^2} < 1$, system (7.4-7.6) has no feasible solution. Now (7.12) gives $r \leq \frac{1}{k}$. To complete the example, we show that the observations violate GARP, since

$$\begin{aligned} p^{(1)} \cdot x^{(2)} &= k^2 \leq b \\ p^{(2)} \cdot x^{(1)} &= 1 \leq b. \end{aligned}$$

Clearly, for any $e < 1$ the observations $D(e)$ satisfy GARP, since then $p^{(1)} \cdot x^{(2)} = k^2 > eb = ek^2$. Thus e^* is arbitrarily close to 1, while r^* is at most $\frac{1}{k}$ for any $k > 1$.

Chapter 8

Bounded Rationality in Matching Pennies[1]

In the classical setting of Game Theory, one of the core assumptions is that all participating agents are "fully rational". This amounts not only to the fact that an agent must be able to make optimal decisions, given the other players' actions, but also to the fact that he must understand how these actions will affect the behavior of all other participants. If this is the case, a Nash equilibrium can be viewed as a set of strategies in which each agent is simply computing his best response given his opponents' actions. However, in real world strategic interactions, people often behave in manners that are not fully rational. There are many reasons behind non-rational behavior, we focus on two: limitations on computation and limitations on randomness.

Since the work of Simon [1955], much research has focused on defining models that take computational issues into account. In recent years, the idea that the full rationality assumption is often unrealistic has been formalized using tools and ideas from computational complexity. It is in fact easy to come up with settings, as in Fortnow and Santhanam [2010], in which simply computing a best response strategy involves solving a computationally hard problem. Furthermore there is strong evidence that, in general, the problem of finding a Nash equilibrium is computationally difficult for matrix games (Daskalakis et al. [2009], Chen and Deng [2006]).

Traditionally bounded rationality has focused on two computational

[1] The work presented in this chapter has already appeared in publication, see Budinich and Fortnow [2011].

resources: time and space. In this work we focus on another fundamental resource: randomness.

It is a basic fact that games in which agents are not allowed to randomize might have no Nash equilibrium. In this sense, randomness is essential in game theory. We focus on a simple two player zero-sum game that captures this: Matching Pennies (Figure 8.1). Specifically, we consider the repeated version of Matching Pennies, played for n rounds. In this game, the unique Nash equilibrium is the one in which, at every round, both players choose one of their two strategies uniformly at random. The algorithm that implements this strategy requires n random coins, one for each round. The main question we address in this chapter is: can there be Nash equilibria if the amount of randomness available to both players is less than n?

		Player 2 (P2)	
		H	T
Player 1 (P1)	H	$1, -1$	$-1, 1$
	T	$-1, 1$	$1, -1$

Figure 8.1: The payoff bimatrix for the Matching Pennies game.

First we show that, in general, the game cannot have a Nash equilibrium in which both players have only a fraction of n random coins. In particular, when we give both players $n(1-\gamma)$ random coins, we can only achieve a γ-Nash equilibrium. This turns out to be tight, in the sense that we can show that any game with a γ-Nash equilibrium both players must have at least $n(1-\gamma)$ coins.

The proof of this fact, however, relies on the players' ability to implement a strategy that runs in exponential time. We then consider games in which the players' strategies are polynomially-bounded. Using ideas developed in cryptography and computational complexity we show that, in this setting, ε-Nash equilibria that use only n^{δ} coins exist if and only if one-way functions exist.

We also show that the amount of randomness can be "traded" for time. If we allow one of the players to run in arbitrary polynomial time, but use only $O(\log n)$ bits, we can still achieve a ε-Nash equilibrium if we restrict his opponent to run in time n^k for some fixed $k > 0$, while giving him n^{δ} random bits.

Finally we consider an infinitely repeated game with time discounted utilities. In this case, in general, for any discount factor δ and approximation ε, we can always achieve a ε-Nash equilibrium with only n random coins, if n is large enough. When we limit players' strategies to polynomial size circuits, we can reduce the amount of randomness to n^{δ}, for any $\delta > 0$.

Related Work

There are many recent approaches to bounded rationality using a computational complexity perspective. For instance Halpern and Pass [2010] study games in which players' strategies are Turing machines. The idea of considering randomness as a costly resource in game theory has received only limited attention. Kalyanaraman and Umans [2007] study zero-sum games, and give both an efficient deterministic algorithm for finding ε-Nash equilibria, as well as a weaker, but more general, result in the spirit of our Lemma 8.11, giving a randomness-efficient adaptive on-line algorithm for playing repeated zero-sum games. Hu [2010] also considers a similar setting but he is concerned with computability rather than complexity. He considers infinitely repeated plays of 2 player zero-sum games that have no pure strategy Nash equilibrium, and in which players have a set of feasible actions, which represents both the strategies they can play and the strategies they can predict. In this setting Hu gives necessary and sufficient conditions for the existence of Nash equilibria. Finally Gossner and Tomala [2008], give entropy bounds on Bayesian learning in a game theoretic setting, in a more general framework then this chapter. Their results applied to Matching Pennies do not achieve the tight bounds we get in Lemma 8.11.

The rest of this chapter is organized as follows. In Section 8.1 we introduce the notation and known results used. Section 8.2 presents an information theoretic impossibility result. Section 8.3 considers players whose strategies are limited to polynomial sized Boolean circuit families, while Sections 8.4 and 8.5 give extensions of the main results to complexity pseudorandom number generators and infinitely repeated versions of the game.

8.1 Background and Definitions

8.1.1 Game Theory Notation

Throughout this chapter we consider a repeated game of Matching Pennies. We focus on this game because it captures one of the fundamental aspects of randomness in game theory. Studying such a simple game also allows us to get tight bounds. However, variations of our results extend to other similar 2 person zero-sum repeated games.

The payoffs at each round are shown in Figure 8.1. Let $h : \{\text{H,T}\} \times \{\text{H,T}\} \to \{-1, 1\}$, be the payoff to Player 1 (P1), and $-h$ the payoff for P2. When we allow the players to randomize, we denote as S_i a randomized strategy on $\Delta\{H, T\}$ for player i. P1's expected payoff in one round is

$$\mathbf{E}\left[h(S_1, S_2)\right] = \sum_{s_1, s_2 \in \{\text{H,T}\}} \Pr\left(S_1 = s_1\right) \Pr\left(S_2 = s_2\right) h(s_1, s_2).$$

Let $u : \{\text{H,T}\}^n \times \{\text{H,T}\}^n \to \{-n, \cdots, n\}$ be P1's cumulative payoff when the game is played for n rounds. In the repeated game, mixed strategies can be viewed as distribution over sequences of length n. Let $R_i = (r_i^1, \cdots, r_i^n) \in \Delta\{\text{H,T}\}^n$ denote a randomized strategies for player i. P1's expected cumulative payoff is

$$\mathbf{E}\left[u(R_1, R_2)\right] = \sum_{t=1}^{n} \mathbf{E}\left[h(r_1^t, r_2^t)\right].$$

Finally we define the expected average payoff to P1 for the n-round game as

$$\mathbf{E}\left[U(R_1, R_2)\right] = \frac{\mathbf{E}\left[u(R_1, R_2)\right]}{n},$$

and consequently P2's expected payoff is $-\mathbf{E}\left[U(R_1, R_2)\right]$. To denote player's i payoff we will sometimes use the standard notation $\mathbf{E}\left[U(R_i, R_{-i})\right]$, where R_i is player's i mixed strategy and R_{-i} is his opponent's mixed strategy.

Definition 8.1 (Nash equilibrium). *A pair of mixed strategies (R_1, R_2) is a Nash equilibrium for the n-stage Matching Pennies game if, for i $= 1, 2$:*

$$\mathbf{E}\left[U(R_i, R_{-i})\right] \geq \mathbf{E}\left[U(R_i', R_{-i})\right] \quad \textit{for all } R_i' \in \Delta\{H,T\}^n.$$

In some cases we will consider a relaxed notion of equilibrium, namely ε-Nash equilibrium.

Definition 8.2 (ε-Nash equilibrium). *A pair of mixed strategies (R_1, R_2) is a ε-Nash equilibrium for the n-stage Matching Pennies game if, for $i = 1, 2$:*

$$\mathbf{E}\left[U(R_i, R_{-i})\right] \geq \mathbf{E}\left[U(R_i', R_{-i})\right] - \varepsilon \quad \textit{for all } R_i' \in \Delta\{H,T\}^n.$$

8.1.2 Complexity and Pseudorandomness

We give a brief description of the pseudorandomness tools we need for this chapter. For more details we recommend the textbooks of Arora and Barak [2009] and Goldreich [2007].

The model of computation used throughout most of the chapter is based on Boolean Circuits. We consider circuits with AND, OR and NOT gates, and denote by C_n a circuit with n input nodes. A circuit family $\{C_i\}_{i \in \mathbb{N}}$ is an infinite collection of circuits, intuitively one for each input length.

The size of a circuit $|C_n|$ is the number of gates. A circuit family is polynomial sized if there is a $k > 0$ such that, for all n, $|C_n| \leq n^k$. The class of languages recognizable by families of polynomial sized circuits is called P/Poly. Any language that can be decided in polynomial time by a deterministic or randomized Turing machine is also in P/Poly. Formally P $\subseteq$ BPP $\subseteq$ P/Poly.

A function is one-way if it is easy to compute and hard to invert.

Definition 8.3 (One-way Function). *A one-way function is a polynomial-time computable function $f : \{0,1\}^n \to \{0,1\}$ such that, for all polynomial size circuits D and $y = f(x)$ (where x is chosen uniformly at random on $\{0,1\}^n$), $\Pr(f(D(y)) = f(x)) < n^{-c}$ for all $c > 0$ and sufficiently large n.*

Informally, two objects are indistinguishable if no polynomial sized circuit family can tell them apart with noticeable probability.

Definition 8.4 (Indistinguishability). *Let X, Y be two random variables on $\{0,1\}^n$. We say that X and Y are computationally indistinguishable if for every family of polynomial size circuits $\{C_i\}_{i \in \mathbb{N}}$, every $c > 0$ and for sufficiently large n*

$$\Big| Pr\left(C_n\left(X\right) = 1\right) - Pr\left(C_n\left(Y\right) = 1\right) \Big| < \frac{1}{n^c}.$$

A cryptographic pseudorandom number generator (PRNG) is a deterministic algorithm whose output is computationally indistinguishable from the uniform distribution, provided that it's input is truly random. We will denote by U_k a random variable uniformly distributed on $\{0,1\}^k$.

Definition 8.5 (Cryptographic PRNG). *A cryptographic pseudorandom number generator is a deterministic polynomial time algorithm $G : \{0,1\}^{l(n)} \to \{0,1\}^n$, where $l(n) < n$ is a polynomial time computable function, such that $G(U_{l(n)})$ and U_n are computationally indistinguishable.*

There are two basic properties about pseudorandom number generators that we will use. One relates the notion of pseudorandomness to the notion of predictability.

Definition 8.6 (Unpredictable). *Let $G : \{0,1\}^{l(n)} \to \{0,1\}^n$ be a polynomial time algorithm, and $G(x) = (y_1, \cdots, y_n)$. We call G unpredictable if for every family of polynomial size circuits $\{D_i\}_{i\in\mathbb{N}}$, all $c > 0$ and for sufficiently large n*

$$Pr\left(D_i\left(y_1, \ldots, y_{i-1}\right) = y_i\right) \leq \frac{1}{2} + \frac{1}{n^c}.$$

Intuitively a pseudorandom number generator must be unpredictable, otherwise we could easily build a test for it by using the predictor circuits. In 1982 Yao (see Yao [1982]) proved the opposite implication, thus establishing the following theorem.

Theorem 8.7 (Yao's Theorem). *A polynomial time algorithm $G : \{0,1\}^{l(n)} \to \{0,1\}^n$ is unpredictable if and only if G is a pseudorandom number generator.*

Håstad, Impagliazzo, Luby and Levin in 1999 (see Hastad et al. [1999]) showed how to construct pseudorandom number generators with polynomial expansion based on one-way functions.

Theorem 8.8 (PRNG's from one-way functions). *One way functions exist if and only if for every $\delta > 0$ there is a pseudorandom number generator with $l(n) = n^\delta$.*

Cryptographic pseudorandom number generators' main power lies in the ability to fool any polynomial sized adversary, while running in polynomial time. However, in other areas of complexity, such as derandomization, the crucial issue is having a smaller seed.

Definition 8.9 (Complexity PRNG). *A complexity pseudorandom number generator is a $2^{l(n)}$ time computable function $G : \{0,1\}^{l(n)} \to \{0,1\}^n$, such that for any circuit C of size n*

$$\left| Pr\left(C\left(U_{l(n)}\right) = 1\right) - Pr\left(C\left(G\left(U_n\right)\right) = 1\right) \right| < n^{-1}.$$

The essential difference with cryptographic pseudorandom number generators is the order of quantifiers. A cryptographic pseudorandom number generator fools circuits of an arbitrary polynomial size. The complexity pseudorandom number generator fools circuits only of a fixed polynomial size but under the right assumptions requires far fewer random bits.

Impagliazzo and Wigderson [2001] building on a series of papers starting with Nisan and Wigderson [1994] characterize when complexity pseudorandom number generators exist.

Theorem 8.10. *There exists $L \in \mathsf{DTIME}(2^{O(l(n))})$ and $\varepsilon > 0$ such that no circuit of size at most $2^{\varepsilon l(n)}$ can compute L if and only if there exists a complexity pseudorandom number generator with $l(n) = C \log n$ for some $C > 0$.*

8.2 Information Theoretic Bounds

In this section we make no computational assumptions on the players, and show that there can be no Nash equilibrium if we limit the amount of randomness available to both players.

Lemma 8.11. *For any $\gamma \in [0, 1]$, if P2 has less than $n(1-\gamma)$ random bits, then P1 has a randomized strategy A that achieves an expected average payoff of at least γ.*

Proof. We will give a strategy $A = (a^1, \cdots, a^n)$ for P1 that achieves a high payoff against any strategy $B = (b^1, \cdots, b^n)$ from P2.

Player 1 will enumerate all of P2's possible coin flips, and will obtain a set of $2^{n(1-\gamma)}$ possible strategies, one of which is the one being used by P2. After each play by P2, P1 can eliminate all the strategies that do not play that action at that round. Let S^t be the set of strategies that are consistent with P2's plays up to round t. Initially the set S^1 contains $2^{n(1-\gamma)}$ strategies, and, for all t, $|S^{t+1}| \leq |S^t|$.

The strategy A for P1 is straightforward: at round t, P1 will play based on the most likely event: he will consider all strategies in S^t and play H if the majority of strategies in S^t use H at round t and play T otherwise. Let p^t be the exact fraction of strategies that are the majority at round t,

$$p^t = \frac{\max\{|\{b^t \mid b^t = H\}|, |\{b^t \mid b^t = T\}|\}}{|S^t|}, \tag{8.1}$$

so that $p^t \in [1/2, 1]$. P1's expected payoff at round t is:

$$\mathbf{E}\left[h(a^t, b^t)\right] = p^t - (1 - p^t) = 2p^t - 1 \geq 0.$$

Thus P1's average expected payoff is at least 0. To show that P1 can actually achieve an average expected payoff of γ we need to consider the amount of information P1 gains at each round. We define the following potential function $\phi : \{1, \ldots, n\} \to \mathbb{R}$:

$$\phi(t) = \sum_{k=1}^{t-1} h(a^k, b^k) - \log |S^t|,$$

which considers both the accumulated payoff for P1 and the log-size of the set of consistent strategies. At time $t = 1$ there are $2^{n(1-\gamma)}$ possible strategies for P2, so $\phi(1) = -n(1-\gamma)$. We will now lower bound the expected increase in ϕ at each round. We can express this as

$$\begin{aligned}
\mathbf{E}\left[\phi(t+1) - \phi(t)\right] &= \left(\mathbf{E}\left[\sum_{k=1}^{t} h(a^k, b^k)\right] - \mathbf{E}\left[\sum_{k=1}^{t-1} h(a^k, b^k)\right]\right) \\
&\qquad - \left(\mathbf{E}\left[\log |S^{t+1}|\right] - \mathbf{E}\left[\log |S^t|\right]\right) \\
&= \mathbf{E}\left[h(a^t, b^t)\right] - \left(\mathbf{E}\left[\log |S^{t+1}| - \log |S^t|\right]\right) \\
&= 2p^t - 1 - \left(\mathbf{E}\left[\log |S^{t+1}| - \log |S^t|\right]\right).
\end{aligned}$$

Now consider $\mathbf{E}\left[\log |S^{t+1}|\right]$. When P1 looses he can eliminate a p^t fraction of strategies, thus the new set S^{t+1} will contain a $(1 - p^t)$ fraction of the strategies in S^t. On the other hand, when P1 wins, $|S^{t+1}| = p^t |S^t|$. To complete the analysis we have to consider two cases, since if $p^t = 1$ then $\mathbf{E}\left[\log |S^{t+1}|\right]$ is not well defined.

First assume $p^t = 1$. This happens when all feasible strategies for P2 have the same action at round t. In this case P1 will win with probability 1, and the size of the set of feasible strategies will stay the same. So, overall, the increase in ϕ will be 1.

Now assume $p^t \in [1/2, 1)$. Then, the expected size of the set S^{t+1} is:

$$\mathbf{E}\left[\log |S^{t+1}|\right] = \left((1 - p^t) \log(1 - p^t)|S^t| + p^t \log p^t |S^t|\right). \tag{8.2}$$

The expected change in $\log |S^t|$ does not depend on S^t, but only on p^t, since

$$\begin{aligned}
\mathbf{E}\left[\log |S^{t+1}| - \log |S^t|\right] &= \left[(1 - p^t) \log(1 - p^t)|S^t|\right. \\
&\qquad \left. + p^t \log p^t |S^t|\right] - \log |S^t| \\
&= (1 - p^t) \log(1 - p^t) + p^t \log p^t.
\end{aligned}$$

So that the overall change in potential when $p^t < 1$ is

$$\mathbf{E}\left[\phi(t+1) - \phi(t)\right] \geq 2p^t - 1 - (1-p^t)\log(1-p^t) - p^t \log p^t,$$

which is always at least 1 for $p^t \geq 1/2$.

So, for all $p^t \in [1/2, 1]$, at each step the potential function φ increases by at least 1. Thus, after n rounds we have that

$$\begin{aligned}\mathbf{E}\left[\phi(n)\right] &\geq \phi(1) + \min_t\{n\mathbf{E}\left[\phi(t+1) - \phi(t)\right]\} \\ &\geq -n(1-\gamma) + n = n\gamma.\end{aligned}$$

Since P1's expected payoff is at least $\mathbf{E}\left[\phi(n)\right]$, this completes the proof. □

This result immediately implies that, without any computational assumption, there can be no equilibrium with less than n random coins.

Corollary 8.12. *For all $\gamma_1, \gamma_2 \in [0,1]$ such that $\gamma_1 + \gamma_2 > 0$, if P1 and P2 have, respectively, $n(1-\gamma_1)$ and $n(1-\gamma_2)$ random coins, then there can be no Nash equilibrium in the n-stage Matching Pennies repeated game.*

Proof. Assume, by contradiction, that (S_1, S_2) is such a Nash equilibrium. By Lemma 8.11, P1's expected payoff must be $\mathbf{E}\left[U(S_1, S_2)\right] \geq \gamma_2$ and P2's payoff $-\mathbf{E}\left[U(S_1, S_2)\right] \geq \gamma_1$, otherwise they would be better off by using the majority strategy in the proof of Lemma 8.11. Summing the two inequalities we get $\gamma_1 + \gamma_2 \leq 0$, a contradiction, since we assume $\gamma_1 + \gamma_2 > 0$. □

However, if we limit the amount of randomness available to both players, we are still able to achieve an ε-Nash equilibrium. Furthermore, if the game has a ε-Nash equilibrium, then both players must have at least $(1-\varepsilon)n$ random coins.

Theorem 8.13. *Let $\gamma \in [0,1]$. The game has a γ-Nash equilibrium if and only if both players have $n(1-\gamma)$ random coins.*

Proof. To show the "only-if" implication, consider, by way of contradiction, a game that has a γ-Nash equilibrium (S_1, S_2) but in which both players have less than $n(1-\gamma)$ random bits. Thus there must be a $\gamma' > \gamma$ such that they have exactly $n(1-\gamma')$ random bits. Since (S_1, S_2) is a γ-Nash equilibrium, it must be the case that, for any strategy S_1' for P1

$$\mathbf{E}\left[U(S_1, S_2)\right] \geq \mathbf{E}\left[U(S_1', S_2)\right] - \gamma.$$

By Lemma 8.11 we know that both players have a strategy that achieves a payoff of at least $\gamma' > \gamma$, so that the above implies $\mathbf{E}[U(S_1, S_2)] > 0$. Applying the same argument to P2, we get $-\mathbf{E}[U(S_1, S_2)] > 0$, a contradiction.

The "if" part follows from Lemma 8.14 below. □

Lemma 8.14. *Let $\gamma \in [0, 1]$. If both players have $n(1-\gamma)$ random coins, then the game has a γ-Nash equilibrium.*

For simplicity we assume γn is even.

Proof. Consider the following strategies: both player use their random coins to play uniformly at random for the first $n(1-\gamma)$ rounds. Thereafter P1 will always play H, while P2 will alternate between H and T, playing $H, T, \ldots$. We claim that this is a γ-Nash equilibrium.

First notice that no player can improve his payoff in the first $n(1-\gamma)$ rounds, given his opponent's strategy. Let's consider the remaining $n\gamma$ rounds. P1 could improve his payoff by playing $H, T, H, T \ldots$, however this only increases his payoff by γ. This holds also for P2, that could play $T, T, \ldots$, however gaining only γ. □

8.3 Computationally Efficient Players

The proof of Lemma 8.11 in the previous section relies heavily the fact that we make no computational assumptions. In particular, to implement the majority strategy and compute p^t in (8.1) requires solving #P hard problems. If we restrict the players to run in time polynomial in n this particular strategy likely becomes unfeasible. In this setting, under reasonable complexity assumptions, it is possible to greatly reduce the amount of randomness and, at the same time, achieve a ε-Nash equilibrium.

We consider players' whose actions are polynomial size Boolean circuits. A strategy is thus a circuit family $\{C_i\}_{i \in \mathbb{N}}$, such that circuit $C_{l(n)}$ takes as input $l(n)$ random coins and outputs the n actions to be played. Notice that this definition implies that each agent can simulate any of his opponent's strategies.

We consider equilibria that use n^δ random coins for any $\delta > 0$. Theorem 8.15 shows that such ε-Nash equilibria exist if and only if one-way functions exist.

Theorem 8.15. *For all $\varepsilon, \delta > 0$ and sufficiently large n, ε-Nash equilibria that use only n^δ random coins exist, where $\varepsilon = n^{-k}$ for all $k > 0$ and sufficiently large n's, if and only if one-way functions exist.*

Proof. The if part is Lemma 8.17, while the only-if part is Lemma 8.18. □

As a preliminary result we show that, in our setting, the expected utility when at least one player uses a pseudorandom number generator can't be too far from the expected utility when playing uniformly at random.

Lemma 8.16. *Assume one-way functions exist, and let G be the strategy corresponding to the output of a pseudorandom number generator. For any strategy S, for all $k > 0$ and sufficiently large n,*

$$\left|\mathbf{E}\left[U(G,S)\right]\right| \le n^{-k} \textit{ and } \left|\mathbf{E}\left[U(S,G)\right]\right| \le n^{-k}.$$

Proof. We prove only the first inequality, the proof for the second one being symmetric.

Proof by contradiction. Assuming there is a $k > 0$ such that $\left|\mathbf{E}\left[U(G,S)\right]\right| > n^{-k}$ for infinitely many n's, we will construct a test T for G, and show that

$$\left|\Pr\left(T\left(G\left(U_{l(n)}\right)\right) = 1\right) - \Pr\left(T\left(U_n\right) = 1\right)\right| > n^{-c} \tag{8.3}$$

for some $c > 0$ and infinitely many n's, thus contradicting the assumption that G is a pseudorandom number generator.

First consider the n random variables $(A_1(C_1,C_2), \ldots, A_n(C_1,C_2))$, where $A_i(C_1,C_2)$ is simply P1's payoff. Since $\sum_{i=1}^{n} A_i(G,S) = nU(G,S)$,

$$\sum_{i=1}^{n} \mathbf{E}\left[A_i(G,S)\right] > n^{1-k}.$$

This implies that there must be an i such that $\mathbf{E}\left[A_i(G,S)\right] > n^{-k}$. Fix that i.

The test T takes as input an n-bit sequence x and generates a sequence of plays s according to strategy S. Now T simulates an n-stage repeated Matching Pennies game with strategies (x,s). If P1 wins the i-th round then it will output 1, otherwise the output will be 0. In other words, T outputs 1 if and only if $A_i(x,s) = 1$.

When x is drawn from the uniform distribution, P1 will win with probability $1/2$, or $\Pr\left(T(U_n) = 1\right) = 1/2$.

Now notice, that since $A_i \in \{-1,1\}$, $\Pr\left(A_i = 1\right) = \frac{\mathbf{E}[A_i]+1}{2}$. This implies that

$$\Pr\left(T(G(U_{l(n)})) = 1\right) = \frac{\mathbf{E}\left[A_i\right]+1}{2} > \frac{1}{2n^k} + \frac{1}{2}.$$

Thus

$$\left|\Pr\left(T\left(G_1\left(U_{l(n)}\right)\right) = 1\right) - \Pr\left(T\left(U_n\right) = 1\right)\right| > \frac{1}{2n^k},$$

which proves the lemma. □

Lemma 8.17. *If one-way functions exist, then for every $\delta, k > 0$ and for sufficiently large n, the n-stage has an n^{-k}-Nash equilibrium in which each player uses at most n^δ random bits and runs in time polynomial in n.*

Proof. Assume, by contradiction, that one-way functions exist but there are values $\delta > 0$ and $k > 0$ such that the game has no n^{-k}-Nash equilibrium in which players use at most n^δ random coins.

Since we assume one-way functions exist, by Theorem 8.8 there exist pseudorandom number generators that use n^δ coins. Assume both players use the output of such pseudorandom number generators as their strategies (which we call, respectively, G_1 and G_2). Since we are assuming that this is not a n^{-k}-Nash equilibrium, one of the players, say Player 1, must have a strategy A such that

$$\mathbf{E}\left[U\left(A, G_2\right)\right] > \mathbf{E}\left[U\left(G_1, G_2\right)\right] + n^{-k}. \tag{8.4}$$

By Lemma 8.16 we can choose a $k' > 0$ such that

$$\mathbf{E}\left[U\left(A, G_2\right)\right] > -n^{-k'} + n^{-k}. \tag{8.5}$$

Pick $c = k' = k + 1$, so that $\mathbf{E}\left[U\left(A, G_2\right)\right] > n^{-c}$ for $n > 2$. This contradicts Lemma 8.16, proving the claim. □

We now prove the opposite direction, that is that the existence of Nash equilibria that use few random bits implies the existence of one-way functions.

Lemma 8.18. *If for every $\delta > 0$ there is a Nash equilibrium in which each player uses n^δ random bits and runs in polynomial time, then one-way functions exist.*

Proof. Let (A, B) be such a Nash equilibrium and assume, by contradiction, that one-way functions don't exist. This implies that pseudorandom number generators can't exist (Goldreich [2007]), and so, A and B can't be sequences that are computationally indistinguishable from uniform.

Thus, by Yao's theorem (Theorem 8.7), we know that there are polynomial size circuit families $\{C_i\}_{i\in\mathbb{N}}$ and $\{D_i\}_{i\in\mathbb{N}}$ such that

$$\begin{aligned}\Pr\left(C_i(y_1,\ldots,y_i) = y_{i+1}\right) &> 1/2+\delta_1\\ \Pr\left(D_i(z_1,\ldots,z_i) = z_{i+1}\right) &> 1/2+\delta_2,\end{aligned}$$

for some $\delta_1,\delta_2 > 0$, where $A(x_1,\ldots,x_{l_1(n)}) = (y_1,y_2,\ldots,y_n)$ and $B(x_1,\ldots,x_{l_2(n)}) = (z_1,z_2,\ldots,z_n)$.

To get a contradiction it is sufficient to show that players are better off by using the predictor circuits C and D. Consider Player 1: using D, at each round he can guess, given the previous history, the opponent's next move with probability $\frac{1}{2}+\delta_1$. Thus his expected payoff at any round t is

$$\mathbf{E}\left[h(d^t,b^t)\right] > \frac{1}{2}+\delta_1-\frac{1}{2}+\delta_1 = 2\delta_1,$$

where the expectation is over the internal coin tosses of the predictor circuit D. The overall expected payoff is

$$\mathbf{E}\left[U(D,B)\right] > \frac{1}{n}\sum_{t=1}^{n} 2\delta = 2\delta_1.$$

Now, let $w = \mathbf{E}\left[U(A,B)\right]$ be the value of the expected payoff when players play (A,B). Consider the following cases:

i) $w \le 0$: this implies that Player 1 could gain $2\delta_1$ by using strategy D,

ii) $w > 0$: by definition Player 2's expected payoff is $-\mathbf{E}\left[U(A,B)\right] < 0$, so Player 2 can achieve a higher payoff by using his predictor circuit C,

In both cases we see that (A,B) can't be a Nash equilibrium, a contradiction. □

8.4 Exchanging Time for Randomness

In this section we determine conditions under which a ε-Nash equilibrium can arise, given that one of the players has only a logarithmic amount of randomness and his opponent must run in time n^k for some fixed k. This shows how we can trade off randomness for time; the player with $O(\log n)$ random bits runs in time polynomial in n, while the player with more random bits runs in fixed polynomial time.

Theorem 8.19. *Assume there exists $f \in DTIME(2^{O(l(n))})$ and $\varepsilon > 0$ such that no circuit of size at most $2^{\varepsilon l(n)}$ can compute f and that one-way functions exist. Let Player 1's strategies be circuits of size at most n^k that use at most n^δ random bits for some $k > 2 + c\delta$, where $c \geq 1$ is a constant related to the implementation of a cryptographic pseudorandom number generator. Assume Player 2 has access to only $M \log n$ random bits. As long as $M > Ck$, where C is the constant in Theorem 8.10, then for all $\varepsilon > 0$ and sufficiently large n there is a ε-Nash equilibrium.*

Proof. Let G_1 be the cryptographic pseudorandom number generator available to Player 1 and G_2 be the complexity pseudorandom number generator used by player 2. Furthermore let $\mathcal{S}_1$ be the set of all possible strategies for P1 (for all $\delta > 0$ circuits of size at most n^k that use n^δ random bits), and $\mathcal{S}_2$ the set of strategies available to P2 (polynomial size circuit families and $M \log n$ random coins). We will show that for all $\varepsilon > 0$ and sufficiently large n, (G_1, G_2) is a ε-Nash equilibrium with the required properties.

First we argue that, for all $\gamma > 0$ and sufficiently large n, $|\mathbf{E}[U(G_1, G_2)]| < \gamma$. The proof of this fact is similar to the proof of Lemma 8.16, showing by way of contradiction, that if $|\mathbf{E}[U(G_1, G_2)]| \geq \gamma$ then we can build a test for the cryptographic pseudorandom number generator G_1.

Now we show that, for the appropriate setting of parameters, G_1 fools $\mathcal{S}_2$ and G_2 fools $\mathcal{S}_1$. For any k, Player 2 can fool circuits of size n^k by using $C \log n^k = Ck \log n$ random bits. So, for $M > Ck$, G_2 fools $\mathcal{S}_1$. Notice also that since Player 2 runs in time $O(n^{Ck})$, the cryptographic pseudorandom number generator G_1 fools $\mathcal{S}_2$. Let h be the one-way permutation used by the pseudorandom number generator G_1, and assume h us computable in time n^c for some $c > 0$. Given h, G_1 is defined as follows: let $x, y \in \{0,1\}^{\frac{n^\delta}{2}}$, and let (x, y) be G_1's seed (notice that $|(x, y)| = n^\delta$), then

$$G_1(x, y) = \left(f^n(x) \odot y, f^{n-1}(x) \odot y, \ldots, f(x) \odot y\right),$$

where $x \odot y = \sum_i x_i y_i \mod 2$. There are $O(n^2)$ applications of h, so G_1 runs in time $O(n^{2+c\delta})$. So, for $k \geq 2 + c\delta$, G_1 fools $\mathcal{S}_2$.

At this point we're almost done. As in Lemma 8.17 assume, by contradiction, that the assumptions in the theorem hold but (G_1, G_2) is not a ε-Nash equilibrium for some $\varepsilon > 0$. This implies that at least one of the two players can improve his expected payoff by more than ε by switching to some other strategy. First consider P2, and assume there is a strategy $S_2 \in \mathcal{S}_2$ such that

$$\mathbf{E}[U(G_1, S_2)] > \mathbf{E}[U(G_1, G_2)] + \varepsilon.$$

As in Lemma 8.17 this implies that S_2 would be a test for the cryptographic pseudorandom number generator G_1, contradicting the fact that G_1 fools $\mathcal{S}_2$. Similarly, assume P1 has a strategy $S_1 \in \mathcal{S}_1$ such that

$$\mathbf{E}\left[U(S_1, G_2)\right] > \mathbf{E}\left[U(G_1, G_2)\right] + \varepsilon.$$

Again, S_1 can easily be made into a test for G_2, contradicting the fact that G_2 fools $\mathcal{S}_1$. □

8.5 Infinite Play

We now consider an infinitely repeated game of Matching Pennies, and show that, if utilities are time discounted, we can always achieve a ε-Nash equilibria using a large enough (but finite) amount of random coins. First we determine the least amount of randomness required to achieve a ε-Nash equilibrium in the general, i.e. computationally unbounded, case.

Lemma 8.20. *For all discount factors $\delta \in (0, 1)$ and all $\varepsilon > 0$, there is an ε-Nash equilibrium in which the players use only n random bits, for $n > \frac{\log \varepsilon(1-\delta)}{\log \delta}$.*

Proof. Given $\delta, \varepsilon > 0$ consider the following strategies: both players play the Nash equilibrium strategy for the first n rounds. After this P1 will always play H, while P2 will play H and T alternatively. The overall expected payoff is 0. However, after round n, both players could switch to a strategy that always wins, achieving a total expected payoff of $0 + \sum_{t=n}^{\infty} \delta^t = \frac{\delta^n}{1-\delta}$. To ensure that our strategies are indeed a ε-Nash equilibrium we just need to make sure that

$$0 > \frac{\delta^n}{1-\delta} - \varepsilon.$$

Rearranging and taking logarithms we get $n > \frac{\log \varepsilon(1-\delta)}{\log \delta}$. □

Now we consider players' whose strategies are families of polynomial size Boolean circuits (as in Section 8.3), and assume one-way functions exist. We first give a version of Lemma 8.16 for time discounted utilities on a finite number of rounds.

Lemma 8.21. *Assume one-way functions exist, and let $G = (g^1, \ldots, g^n)$ be the strategy corresponding to the output of a cryptographic pseudorandom* *number generator. Let $S = (s^1, \ldots, s^n)$ be any strategy. For all $\delta \in (0,1)$, $k > 0$ and for sufficiently large n*

$$\left| \mathbf{E}\left[\sum_{t=1}^{n} \delta^t h(g^t, s^t) \right] \right| \leq n^{-k} \textit{ and } \left| \mathbf{E}\left[\sum_{t=1}^{n} \delta^t h(s^t, g^t) \right] \right| \leq n^{-k}.$$

Proof. Again we give the proof only for the first inequality. Assume, by contradiction, that $\left|\mathbf{E}\left[\sum_{t=1}^{n} \delta^t h(g^t, s^t)\right]\right| > n^{-k}$ for some δ, k and infinitely many n's. Consider the random variables $A_1(c_1^1, c_2^1), \ldots, A_n(c_1^n, c_2^n)$, defined as $A_t(c_1^t, c_2^t) = 1$ if P1 wins round t when playing according to (C_1, C_2) and 0 otherwise. Let $A(C_1, C_2) = \sum_t \delta^t A_t(c_1^t, c_2^t)$, so that $\mathbf{E}\left[A(G,S)\right] \geq \left|\mathbf{E}\left[\sum_{t=1}^{n} \delta^t h(g^t, s^t)\right]\right| > n^{-k}$. This implies that there is a t such that $\delta^t \mathbf{E}\left[A_t(g^t, s^t)\right] > n^{-k-1}$, which implies $\mathbf{E}\left[A_t(g^t, s^t)\right] > n^{-k-1}$. Fix that t. As in Lemma 8.16, consider the test T that, given a sequence of plays x, generates a play s from S and outputs 1 if P1 wins round t and outputs 0 otherwise.

When x is drawn uniformly at random, $\Pr(T(U_n) = 1) = 1/2$. On the other hand, when x is G's output,

$$\Pr\left(T(G(U_{l(n)})) = 1\right) = \mathbf{E}\left[A_t\right] > n^{-k-1}.$$

Now

$$\left|\Pr\left(T(G(U_{l(n)})) = 1\right) - \Pr\left(T(U_n) = 1\right)\right| > \frac{1}{2} - \frac{1}{n^{k+1}} = \frac{1}{n^c},$$

for $c > -\frac{\log(1/2 - n^{-1-k})}{\log n} \geq 0$, contradicting the assumption that G is a pseudorandom number generator. □

Using the above Lemma we can show that, for all discount factors, we can greatly reduce the amount of random coins needed to get an ε-Nash equilibrium.

Lemma 8.22. *For all discount factors $\delta \in (0,1)$, all $\varepsilon > 0$ and all $\xi > 0$, there is a n^{-k}-Nash equilibrium in which players use only n^{ξ} random coins, for sufficiently large n's.*

Proof. As in the proof of Lemma 8.20 we consider the following strategy for both players: for the first n rounds play the output of a cryptographic pseudorandom number generator G, with seed length n^{ξ}. Thereafter P1 will always play H, while P2 will alternate between H and T. Pick any $k > 0$, we now show that this is a n^{-k}-Nash equilibrium. By Lemma 8.21 we can pick $k' = k/2$ such that the expected utility in the first n

rounds lies in the interval $[-n^{-k'}, n^{-k'}]$. To ensure that this is a n^{-k}-Nash equilibrium we just need to show that

$$-n^{-k'} > n^{-k'} + \frac{\delta^n}{(1-\delta)} - n^{-k},$$

or

$$-2n^{-k'} + n^{-2k'} > \frac{\delta^n}{(1-\delta)}.$$

Now, for any $c > 0$, if we set $k' = \frac{\log((\sqrt{c+1}-1)/c)}{\log n}$, then the left hand side of the above inequality is c, so that it always holds for sufficiently large n's. □

Thus, given any n that satisfies the conditions in Lemma 8.20, there can be a n^{-k}-Nash equilibrium using n^{δ} coins, for any $\delta > 0$. To see this, consider an m sufficiently large so that Lemma 8.22 holds and pick $\xi = \delta \frac{\log n}{\log m}$.

8.6 Conclusions

We have shown how, in a simple setting, reducing the amount of randomness available to players affects Nash equilibria. In particular, if we make no computational assumptions on the players, there is a direct tradeoff between the amount of randomness and the approximation to a Nash equilibrium we can achieve. If, instead, players are bound to run in polynomial time, we can get very close to a Nash equilibrium with only n^{δ} random coins, for any $\delta > 0$.

Some directions for future research include:

- Is it possible to extend Lemma 8.11 to m player games, for $m > 2$? Notice that the strategy used in that proof does not generalize to this setting.
- Under what circumstances is it possible to further reduce the amount of randomness available (say to $O(\log n)$ for both players)?
- Is it possible to extend these results to general zero-sum games or even non zero-sum games?

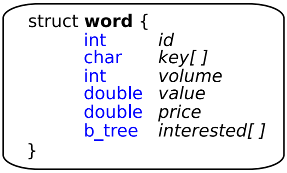
Part V

Appendix

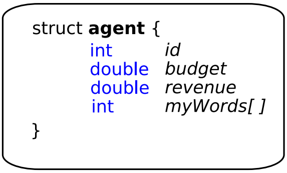
Appendix A

Implementation Details

Agents and words are represented by simple data structures, and are stored in memory in plain arrays.

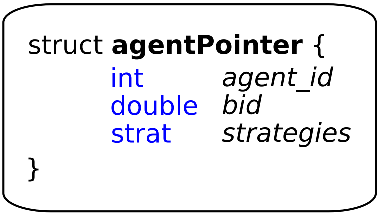


```
struct word {
        int       id
        char      key[ ]
        int       volume
        double    value
        double    price
        b_tree    interested[ ]
}
```



```
struct agent {
        int       id
        double    budget
        double    revenue
        int       myWords[ ]
}
```

The bids, which are potentially as many as mn (where m is the number of bidders and n the number of words), are stored in a different data structure, called `agentPointer`.



```
struct agentPointer {
        int       agent_id
        double    bid
        strat     strategies
}
```

This basically contains the bid itself (among an `originalBid`) and a pointer (actually an `int` corresponding to the correct array entry) that connects the bid with the agent that made it. Each word has a pointer to an AVL tree, in which it stores all the `agentPointers` related to it. In this way, when a query is made, we simply retrieve the first elements of the tree, and can directly access the relative agents.

Each tree basically contains a list of the agents interested in it. The simulator, through a compile-time setting, gives the possibility to store also the reverse information. In this case each agent stores a pointer to a list of all the words he is interested in. This clearly increases the memory usage, and is potentially worse than storing the full mn entries (the worst case is the one in which every agent bids on every keyword; in this case the implementation just described would store $2mn$ entries). Nonetheless, the data is very sparse, and so usually storing both lists is still better than storing the whole matrix. Even though this kind of implementation occupies nearly twice the memory, it greatly improves the speed in certain routines. The proposed solutions seem an acceptable tradeoff: if we are interested in keeping memory usage to a minimum, we can do so by storing just the AVL trees (and this is the minimum possible amount of memory). On the other hand, if we are interested in performance, we store both lists.

Dependencies The simulator, aside from the standard `C` libraries, has the following dependencies

`GNU libavl`
: available on-line,[1] is heavily used, since each tree stores the corresponding `agentPointers` in a `pavl_table`, which is an AVL tree with parent pointers. The library has been slightly modified (files `src/lib/m_pavl.c` and `src/lib/m_pavl.h` in the source tree), and is now included in the source of the simulator (so it is not an actual dependency).

`zlib`
: available on-line,[2] used just when compressing (and decompressing) the saved files. Usually installed in most systems.

`GNU GSL`
: the GNU scientific library, available on-line,[3] is used primarily for sensible pseudo random number generation. It is usually not installed by default, anyways available in most package repositories, and can also very easily be compiled directly from source.

A.1 The Configuration File

The simulator, after having parsed the command line parameters, reads some parameters from a given configuration file (`Config.cfg` by de-

[1] `http://www.stanford.edu/~blp/avl/`

[2] `http://www.zlib.net/`

[3] `http://www.gnu.org/software/gsl/`

fault). The most important variables that can be set are

word-list the name of the word list file

budgets the maximum and minimum values can be set

bids hard limits on maximum/minimum bids

words per agent the maximum number of words each agent can bid on

keyword spreading many parameters can be set here, such as the percentage of agents that are allowed to do keyword spreading, the number of queries after which keyword spreading can start, the probability with which possible word changes are performed, and a hard limit on the number of words a single agent can change. Finally synonyms database can be chosen (for the moment just the one created by clustering and the one derived from Wordnet)

strategies various parameters regarding strategic agents, mainly the number of strategic agents and the type of strategy to be used

Many other options can be set, and the default file contains more details on the variables and possible values.

A.2 Command Line Parameters

The simulator accepts a number of command line parameters. The required ones are

`-a, -agents NUM`
the number of agents to generate

`-q, -queries NUM`
the number of queries.

These arguments are not both mandatory when we are reading/saving data to a file

`-r, -read FILE`
agents will be read from file `FILE`. If this option is specified the `-a NUM` option need not be specified (and if it is it is overridden by the number of agents saved in the file).

`-w, -write FILE`
agents will be saved to the given file. If this option is specified the `-q NUM` option is not needed (in case we simply want to generate and save the agents).

There are then a number of optional flags

`-s, -sort {b,f}`
controls how agents are sorted in the AVL trees. The default option is to sort them by bid (`b`), while `f` uses tradeoff functions for the sort.

`-p, -payment {v,g,f}`
the auction mechanism used. The default is `v`, corresponding to the VCG auction, while `g` uses the GSP mechanism, and `f` is a simple first price auction.

`-c, -change`
keyword spreading will be performed

`-t, -strategies`
strategies will be performed.

`-o, -output FILE`
the name of the log file (placed in the `runs/` subdirecotory). The default value is `graph.txt`.

`-config FILE`
the name of the configuration file. The default value is `Config.cfg`.

There are other parameters that control features yet to be implemented (or checked).

A.3 Notable Functions

`main()` The main function simply parses the configuration file (see A.1) and then calls the `init()` function, which in turn initializes (in the correct order!) all the necessary data structures.

`init()` This function that initializes most of the data structures used by the program. The order in which it calls the specific initialization functions is important. Usually there are two initializations per component: in the first pass we simply check some parameters and possibly seed the pseudo random number generators, while in the second pass we actually allocate the data structures and initialize their components. Based on the command line parameters, it decides if agents will be generated or read from a file.

`generateBidders()` When agents are not read from a file, they are generated on the spot. This procedure basically allocates the agents' array, and cycles through each one of them. It initializes the agent's budget and decides how many keywords he will bid on (according to a Pareto distribution whose parameters change with the number of agents). Then, until he has not reached this number of bids, a random value (according to a different Pareto distribution) is chosen, and if the agent has not yet bid on this word, a bid is generated (according to the word's true value and some agent-specific parameters) and the agent is inserted in the word's AVL tree.

A.3.1 The Query Cycle

`do_a_run()` This function is called directly by the `main()`, and primarily deals with doing all the queries and gathering the log data. Its principal component is a simple loop on the number of queries that, after calling the function that actually does the query, updates the log file and the feedback to the user.

`doQuery()` The first step is to decide which keyword this query corresponds to. To do so, the file containing the word list is scanned, and the key is selected according to the distribution of the estimated search volumes. Then, if there are still agents with a positive budget interested in this word, a click on a particular slot is generated (according to the probabilities defined in the configuration file), and the winner is selected from the word's AVL tree.

`make_him_pay()` Given the winning agent, this function computes (and returns) the payment due by the agent. First it uses an external function, that based on the mechanism chosen (i.e. VCG or GSP), computes the correct payment. Then it updates the agent itself (decrementing his budget and possibly removing him from the simulation if his budget is negative after this payment). As a last step this function updates some global variables, keeping track of the revenues, the number of alive agents and other details.

A.4 Keyword Spreading

When keyword spreading is enabled two most important changes occur:

i) during agent creation (or when agents are read from a saved file), a certain percentage of them (according to parameters in the configuration file) are allowed to be "changing" agents,

ii) at the end of each query (in the `doQuery()` function), the control is passed to the keyword spreading functions. These will determine if the change can be applied and, if that is the case, will manage all necessary calculations and updates.

It is important to note that the keyword spreading property is a characteristic of the single agents.

`will_change_words()` This is called after each agent is generated (or read from a file) and determines, according to the parameters given in the configuration file, if this agent will be allowed to change keywords for their synonyms.

`should_change_words()` This is the function that established whether keyword spreading should be applied in this query. Primarily it checks if a sufficient number of queries has already been done, and if there are any agents in the query's AVL tree that are allowed to change words. Even when these two conditions are met, keyword spreading is applied with a given probability, trying to reflect the fact that it is not a task that is performed very frequently in the real world. When all the conditions are met, the function then chooses (uniformly at random) a "changing" agent.

`change_words()` If there is a set of synonyms that satisfies the requirements, this function performs the actual change. It removes the given agent from the original keyword's AVL tree and inserts him in all the AVL trees for the new words.

`best_knapsack()` A standard dynamic programming algorithm for the weighted knapsack problem. Since the number of synonyms for each word is rather small (see Table 6.1 and Figure 6.1a), the pseudo polynomial running time does not appear a big issue. As described in Section A.4 the maximum weight is the estimated cost of the word being considered for removal, while the value of each synonym is given by the estimated revenue this agent could obtain from it. Since an agent (initially) has a private valuation only for the words he bid on, this procedure must generate private valuations for the synonyms.

Appendix B

Usage and Notes

The simulator, to compile and run correctly, expects a particular directory structure, and the presence of some auxiliary files. The most important ones are reported in Table B.1.

Directories

`src/`	contains all the source code files
`src/lib/`	contains some libraries
`build/`	temporary object files
`runs/`	where the log-files are saved

Files

`wordlist_augmented_utf8.txt`	the default word-list file
`wordnet_synonyms.txt`	synonyms from the Wordnet database
`clustering_synonyms.txt`	synonyms from clustering
`Config.cfg`	default configuration file
`Makefile`	the compiling directions for the `make` program

Table B.1: Directory structure assumed by the simulator.

B.1 A Sample Run

This section documents a run of the simulator. Having compiled and linked the program, we can now call it. This will start a run with 10^5

agents and 10^6 queries. The agents will be generated on the fly, and the mechanism used will be the VCG one (the default).

```
$ ./sim -a1e5 -q1e6
```

The simulator now reports that it has read the configuration file, read the word database and is now generating the agents. It also informs us that it is using the SPEED define, which means that each agent stores a list of the words he's interested in (see Section A).

```
$ ./sim -a1e5 -q1e6
Read Configuration from: Config.cfg
[using SPEED define]
initialized 35867 words...
generating 1.0E+05 agents:
[=====                                   ] \ 17% [17005]
```

Having finished generating the agents the simulator prints out some details of the parameters that have been set (or considered as default), and starts reporting on the queries already completed.

```
$ ./sim -a1e5 -q1e6
Read Configuration from: Config.cfg
[using SPEED define]
initialized 35867 words...
generating 1.0E+05 agents:
<finished agents>
initialized google:
        auction type   : VCG
        sorting method : by bid value
        1.0E+06 total queries
queries done (on 1.0E+06):
[=============                           ] / 36% [360004]
```

Having completed the queries, the simulator simply reports the number of queries actually performed, and the number of agents that still had some budget left over at the end of the run.

```
$ ./sim -a1e5 -q1e6
Read Configuration from: Config.cfg
[using SPEED define]
initialized 35867 words...
```

```
generating 1.0E+05 agents:
<finished agents>
initialized google:
        auction type   : VCG
        sorting method : by bid value
        1.0E+06 total queries
queries done (on 1.0E+06):
<finished queries>
Exiting [done 5.95E+05 queries, 9.43E+04 still alive].
```

B.2 File Formats

The Configuration File

The file is self-documented, and a sample file is included as a comment in the source file `src/readConfig.c`. The parser function, also in `src/readConfig.c`, is `readFile()`.

The Word List

The word-list contains information on the word itself, the estimated number of clicks and the estimated cost per click (as gathered from the Traffic Estimator, see Section 5.1.2). There is a dummy-column (for no specific reason), and the columns are separated by tabs and white spaces. Below are a few sample lines.

```
hotels    116097  -1.0000   2.8500
travel    100647  -1.0000   2.3850
cheap     94174   -1.0000   2.8600
flights   92417   -1.0000   1.8650
```

The parser for the word list is the function `init_words_from_file()`, located in `src/global.c`.

The Synonyms File

Both synonyms file have the same structure, and are simply a comma separated list of words in which the first entry is the term and the successive ones are its synonyms. Spaces are not allowed. Again, a few sample lines follow

```
abroad,study,italian,eu,university
eggs,white,shell,protein,yolk,albumen
aspire,dream,aspiration,ambition,wannabe
```

The parser for both of the synonyms files is the function `parse_clustering()`, located in `src/synonyms.c`.

A partial solution to this problem (adopted in the simulator) is to prohibit that an agent changes the same word multiple times. In this way we kind of "fix" the agent's valuations, since if the first time he rejects the swap, he will never be allowed to consider it again. The only problem left now is the terms that are synonyms of different queries (as illustrated in the example above), which receive two different valuations by the same agent.

Part VI

Conclusions and References

Conclusions and Future Research

The sponsored search problem is still an area of active research, with numerous open problems waiting for a solution. We give here a brief outline of the most interesting ones, and their relation to the work presented in this thesis.

Repeated Auctions From the theoretical perspective, the models that have been studied are still an approximation to what happens in the real world. For instance, the repeated nature of the auctions is a crucial characteristics in the real world, and has received comparatively little attention in the scientific literature. Beyond the models presented in Section 2.7 [page 53], not much has been done, and there is certainly a need for more realistic dynamic models that capture the long-term notions of equilibria.

Information Asymmetry Another characteristic that has received little attention in the classical models is the role of information. Most authors assume that many parameters, such as click through rates and bids, are common knowledge, or, in the Bayesian setting, that detailed distributional information is available. Although this information certainly is present, most of the time the only participant in the market to have accurate estimates is the search engine. The justification that is often given when making the assumption that this information is available also to agents is that, given enough time, they should be able to learn it. However, as illustrated by the experimental results in Section 6.2 [page 105], it might be very costly for an agent to explore the environment by changing his bid, since the clicks he receives while doing so might be more expensive. Furthermore the auctions are repeated numerous times, and the number of participating agents, bids, click through rates and quality scores change constantly over time, making the assumption that an agent might learn these quantities less realistic.

Common Knowledge Even if we assume some agents are able to collect information on the main parameters in the auction, the assumption of common knowledge implies that they also must be sure that all their opponents have the same information, and that these opponents know that they themselves have it, and so on. This is a particularly strong assumption. In particular it would be interesting to see how the properties of the sponsored search markets change when one agent has higher quality information, or is sure some of his opponents have wrong estimates for some parameters.

Bounded Rationality The size, complexity and dynamic nature of these markets make them a prime candidate for the application of bounded rationality. As we have seen in Section 6.1 [page 90], simply improving his set of keywords can yield important gains to an agent.

Mediators Another interesting point is the role of mediators (see Feldman et al. [2010], Ashlagi et al. [2009]). Mediators have a central role in sponsored search markets: there are many companies that administer on-line advertising for different clients. From a mediator's point of view, the market is rather different. For instance a mediator might be seen as an organizer of a bidding ring (see Section 4.1 [page 68]). Thus, besides the interaction of the mediator with the market, there is the internal optimization strategy for the mediator.

Simulations Even from the experimental point of view there is still a lot of work to do. This could potentially benefit both the search engines and the advertisers. Moreover, most of the models hinted at in the previous paragraphs are probably too complex to frame theoretically. In these cases, focused simulation results could point the theoretical research in the right direction. However the biggest problem, as already widely discussed, is the lack of publicly available information. Thus another interesting (and maybe even profitable) research area is the study and development of algorithms that reliably collect and estimate the parameters needed to run simulations, or to directly optimize advertising campaigns.

There are also many other interesting problems in the on-line advertising markets. For instance, search engines such as Google, also sell advertising space on blogs and many other websites. The advertisers buys virtual advertising space on pages related to a certain subject or keywords, and the search engine decides on which pages to display the ads. The owner of the site where the ads are displayed receives a payment either on pay-per-click or on a pay-per-impression basis. This context poses many new problems, from the machine learning problem of matching a specific page to a category or keyword (i.e. to decide if a given page is appropriate for displaying a given ad), to the economics of determining the prices of clicks or impressions. In fact this setting is more complex than the sponsored search one, since there is no fixed number of slots, and the quality of the pages on which ads are displayed might change dramatically, so that clicks might be worth much less, or more, on certain sites.

Another emerging way to sell advertisements on-line are ad Exchanges (see Muthukrishnan [2009]). These are two-sided markets in which search engines (or other content managers) sell advertising space on web pages. As with auctions for sponsored search, the idea is to efficiently set the prices for the space (or clicks), by taking into account the amount of supply and demand.

References

V. Abhishek and K. Hosanagar. Keyword Generation for Search Engine Advertising using Semantic Similarity Between Terms. In *9th ACM Conference on Electronic Commerce - EC*, pages 89–94. ACM, 2007.

S.N. Afriat. The Construction of Utility Functions from Expenditure Data. *International Economic Review*, 8(1):67–77, 1967.

S.N. Afriat. Efficiency Estimation of Production Functions. *International Economic Review*, 13(3):568–598, 1972.

D. Agarwal, A.Z. Broder, D. Chakrabarti, D. Diklic, V. Josifovski, and M. Sayyadian. Estimating Rates of Rare Events at Multiple Resolutions. In *13th ACM International Conference on Knowledge Discovery and Data Mining - SIGKDD*, pages 16–25. ACM, 2007.

G. Aggarwal and S. Muthukrishnan. Theory of Sponsored Search Auctions. In *49th Foundations of Computer Science - FOCS*, pages 7–7. IEEE, 2008.

G. Aggarwal, A. Goel, and R. Motwani. Truthful Auctions for Pricing Search Keywords. In *7th ACM conference on Electronic Commerce - EC*, pages 1–7. ACM, 2006.

G. Aggarwal, J. Feldman, S. Muthukrishnan, and M. Pál. Sponsored Search Auctions with Markovian Users. *Internet and Network Economics*, pages 621–628, 2008.

N. Archak, V.S. Mirrokni, and S. Muthukrishnan. Mining Advertiser-specific User Behavior Using Adfactors. In *19th International Conference on World Wide Web*, pages 31–40. ACM, 2010.

S. Arora and B. Barak. *Computational Complexity: a Modern Approach*. Cambridge University Press, 2009.

I. Ashlagi, D. Monderer, and M. Tennenholtz. Mediators in Position Auctions. *Games and Economic Behavior*, 67(1):2–21, 2009.

L.M. Ausubel and P. Milgrom. The Lovely but Lonely Vickrey Auction. In *Combinatorial Auctions*, pages 17–40. MIT Press, 2006.

M. Babaioff and T. Roughgarden. Equilibrium Efficiency and Price Complexity in Sponsored Search Auctions. In *6th Workshop on Ad Auctions*, 2010.

Z. Bar-Yossef and M. Gurevich. Estimating the Impressionrank of Web Pages. In *18th International Conference on World Wide Web*, pages 41–50. ACM, 2009.

S. Bikhcandani and J. M. Ostroy. From the Assignment Model to Combinatorial Auctions. In P.C. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. The MIT Press, 2006.

A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online Expansion of Rare Queries for Sponsored Search. In *18th International Conference on World Wide Web*, pages 511–520. ACM, 2009.

A.Z. Broder, P. Ciccolo, M. Fontoura, E. Gabrilovich, V. Josifovski, and L. Riedel. Search Advertising using Web Relevance Feedback. In *17th ACM Conference on Information and Knowledge Management*, pages 1013–1022. ACM, 2008.

M. Budinich. Quality scores in sponsored search. Technical report, IIT - CNR Pisa, Italy, 2011.

M. Budinich and L. Fortnow. Repeated Matching Pennies with Limited Randomness. In *ACM Conference on Electronic Commerce - EC*, pages 111–118, 2011.

M. Budinich, B. Codenotti, F. Geraci, and M. Pellegrini. On the Benefits of Keyword Spreading in Sponsored Search Auctions: An Experimental Analysis. In *EC-Web*, pages 158–171, 2010a.

M. Budinich, L. Fortnow, and R. Vohra. Estimating Consumption in the Presence of Errors. Working Paper, 2010b.

M. Budinich, B. Codenotti, F. Geraci, and M. Pellegrini. Estimating Click Through Rates for AdWords using Public Data. In *IEEE CCSIE*, 2011.

M. Cary, A. Das, B. Edelman, I. Giotis, K. Heimerl, A. R. Karlin, C. Mathieu, and M. Schwarz. On Best-Response Bidding in GSP Auctions. Working Paper 13788, National Bureau of Economic Research, February 2008. URL `http://www.nber.org/papers/w13788`.

X. Chen and X. Deng. Settling the Complexity of Two-player Nash Equilibrium. In *47th Foundations of Computer Science - FOCS*, 2006.

M. Ciaramita, V. Murdock, and V. Plachouras. Online Learning from Click Data for Sponsored Search. In *17th International Conference on World Wide Web*, pages 227–236. ACM, 2008.

E.H. Clarke. Multipart Pricing of Public Goods. *Public choice*, 11(1): 17–33, 1971.

N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An Experimental Comparison of Click Position-bias Models. In *International Conference on Web Search and Web Data Mining*, pages 87–94. ACM, 2008.

C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *SIAM Journal of Computing*, 39 (1):195–259, 2009.

K. Dembczynski, W. Kotlowski, and D. Weiss. Predicting AdsâĂŹ Click-through Rate with Decision Rules. In *Workshop on Targeting and Ranking in Online Advertising*, volume 2008. Citeseer, 2008.

I.S. Dhillon, S. Mallela, and R. Kumar. Enhanced Word Clustering for Hierarchical Text Classification. In *8th ACM International Conference on Knowledge Discovery and Data Mining - SIGKDD*, pages 191–200. ACM, 2002.

B. Edelman, M. Ostrovsky, and M. Schwarz. Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords. *American Economic Review*, 97(1):242–259, 2007.

J. Feldman, V.S. Mirrokni, S. Muthukrishnan, and M.M. Pai. Auctions with Intermediaries: Extended Abstract. In *11th ACM Conference on Electronic Commerce - EC*, pages 23–32, 2010.

L. Fortnow and R. Santhanam. Bounding Rationality by Discounting Time. In *1st Innovations in Computer Science - ICS*, 2010.

L. Fortnow and D. Whang. Optimality and Domination in Repeated Games with Bounded Players. In *26th ACM symposium on Theory of Computing*, pages 741–749. ACM, 1994.

A. Fostel, H.E. Scarf, and M.J. Todd. Two New Proofs of AfriatâĂŹs Theorem. *Economic Theory*, 24(1):211–219, 2004.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman & Co., 1979.

A. Gibbard. Manipulation of Voting Schemes: a General Result. *Econometrica: Journal of the Econometric Society*, pages 587–601, 1973.

I. Gluhovsky. Forecasting Click-Through Rates Based on Sponsored Search Advertiser Bids and Intermediate Variable Regression. *ACM Transactions on Internet Technology - TOIT*, 10(3):11, 2010.

O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2007.

R. D. Gomes and K. S. Sweeney. Bayes-Nash Equilibria of the Generalized Second Price Auction. *10th ACM Conference on Electronic Commerce - EC*, page 107âĂŞ108, 2009. doi: 10.1145/1566374.1566389. ACM ID: 1566389.

T.F. Gonzalez. Clustering to Minimize the Maximum Intercluster Distance* 1. *Theoretical Computer Science*, 38:293–306, 1985.

O. Gossner and T. Tomala. Entropy Bounds on Bayesian Learning. *Journal of Mathematical Economics*, 44(1):24–32, 2008. ISSN 0304-4068.

T. Groves. Efficient Collective Choice when Compensation is Possible. *The Review of Economic Studies*, 46(2):227–241, 1979.

J. Y. Halpern and R. Pass. Game Theory with Costly Computation: Formulation and Application to Protocol Security. In *1st Innovations in Computer Science - ICS*, pages 120–142, 2010.

J. D. Hartline. Lectures on Approximation in Mechanism Design. 2011. URL `http://users.eecs.northwestern.edu/~hartline/courses/algorithmic-mechanism-design`.

J. Hastad, R. Impagliazzo, L.A. Levin, and M. Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. ISSN 0097-5397.

D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan, and C. Leggetter. Improving Ad Relevance in Sponsored Search. In *3rd ACM International Conference on Web Search and Data Mining*, pages 361–370. ACM, 2010.

M. Houtman and J. Maks. Determining All Maximal Data Subsets Consistent with Revealed Preference. *Kwantitatieve methoden*, 19:89–104, 1985.

T.-W. Hu. Complexity and Mixed Strategy Equilibria. `http://bit.ly/e4N8cN`, 2010. Working Paper.

R. Impagliazzo and A. Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search. *ACM Transactions on Information Systems (TOIS)*, 25(2), 2007.

P.R. Jordan and M.P. Wellman. Designing an Ad Auctions Game for the Trading Agent Competition. *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, pages 147–162, 2010.

E. Kalai. Bounded Rationality and Strategic Complexity in Repeated Games. *Discussion Papers*, 1987. Northwestern University, Center for Mathematical Studies in Economics and Management Science.

S. Kalyanaraman and C. Umans. Algorithms for Playing Games with Limited Randomness. In *15th Annual European Symposium - ESA*, 2007.

B. Kalyanasundaram and K. R. Pruhs. An Optimal Deterministic Algorithm for Online b-matching. *Theoretical Computer Science*, 233(1): 319–325, 2000.

F. Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8(1):33–37, 1997.

B. Kitts and B.J. LeBlanc. A Trading Agent and Simulator for Keyword Auctions. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 228–235. IEEE Computer Society, 2004.

B. Kitts, P. Laxminarayan, B. Leblanc, and R. Meech. A Formal Analysis of Search Auctions Including Predictions on Click Fraud and Bidding Tactics. In *Workshop on Sponsored Search Auctions*, 2005.

S. Lahaie, D.M. Pennock, A. Saberi, and R.V. Vohra. Sponsored Search Auctions. In *Algorithmic Game Theory* Nisan et al. [2007], pages 699–716.

R. P Leme and ÃL' Tardos. Bayes-Nash Price of Anarchy for GSP. In *AdAuction Workshop 2009*, 2009.

H. Li and N. Abe. Word Clustering and Disambiguation Based on Co-occurrence Data. In *17th International Conference on Computational Linguistics*, pages 749–755. Association for Computational Linguistics, 1998.

A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic theory*. Oxford University Press, 1995.

A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. AdWords and Generalized Online matching. *Journal of the ACM*, 54(5):22, 2007.

G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. Introduction to Wordnet: an On-line Lexical Database*. *International Journal of lexicography*, 3(4):235, 1990.

H. Mizuta and K. Steiglitz. Agent-based Simulation of Dynamic On-line Auctions. In *IEEE Simulation Conference*, volume 2, pages 1772–1777. IEEE, 2000.

S. Muthukrishnan. Ad Exchanges: Research Issues. *Internet and Network Economics*, pages 1–12, 2009.

N. Nisan. Introduction to Mechanism Design (for Computer Scientists). In *Algorithmic Game Theory* Nisan et al. [2007], pages 209âĂŞ–242.

N. Nisan and A. Wigderson. Hardness vs. Randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. ISSN 0022-0000.

N. Nisan, T. Roughgarden, T. Eva, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

M. Ostrovsky and M. Schwarz. Reserve Prices in Internet Advertising Auctions: a Field Experiment. In *Proceedings of the Advertisement Auctions Workshop*, 2010.

C.H. Papadimitriou and M. Yannakakis. On Complexity as Bounded Rationality. In *26th ACM symposium on Theory of computing*, pages 726–733. ACM, 1994.

D. Pardoe, D. Chakraborty, and P. Stone. TacTex09: a Champion Bidding Agent for Ad Auctions. In *9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, pages 1273–1280, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9826571-1-9. URL `http://dl.acm.org/citation.cfm?id=1838206.1838372`.

T.C. Piaw and R.V. Vohra. AfriatâĂŹs Theorem and Negative Cycles. Technical report, 2003. Mimeo.

M. Regelson and D. Fain. Predicting Click-through Rate using Keyword Clusters. In *2nd Workshop on Sponsored Search Auctions*, volume 9623. Citeseer, 2006.

M. Richardson, E. Dominowska, and R. Ragno. Predicting Clicks: Estimating the Click-through Rate for New Ads. In *16th International Conference on World Wide Web*, pages 521–530. ACM, 2007.

K. Roberts. The Characterization of Implementable Choice Rules. *Aggregation and Revelation of Preferences*, pages 321–348, 1979.

A.E. Roth and M.A.O. Sotomayor. *Two-Sided Matching*. Cambridge University Press, 1990.

P. Rusmevichientong and D.P. Williamson. An Adaptive Algorithm for Selecting Profitable Keywords for Search-based Advertising Services. In *7th ACM conference on Electronic commerce*, pages 260–269. ACM, 2006.

M.A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions* 1. *Journal of Economic Theory*, 10(2):187–217, 1975.

B. Shaparenko,
"O. Çetin, and R. Iyer. Data-driven Text Features for Sponsored Search Click Prediction. In *3rd International Workshop on Data Mining and Audience Intelligence for Advertising*, pages 46–54. ACM, 2009.

L. S. Shapley and M. Shubik. The Assignment Game I: the Core. *International Journal of Game Theory*, 1:111–130, 1971. ISSN 0020-7276. URL `http://dx.doi.org/10.1007/BF01753437`. 10.1007/BF01753437.

H.A. Simon. A Behavioral Model of Rational Choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955. ISSN 0033-5533.

M.J. Todd. *The Computation of Fixed Points and Applications*. Springer, 1976.

H.R. Varian. The Nonparametric Approach to Demand Analysis. *Econometrica: Journal of the Econometric Society*, pages 945–973, 1982.

H.R. Varian. Goodness-of-fit for Revealed Preference Tests. *Econometrics*, 1994.

H.R. Varian. Position Auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of finance*, 16(1):8–37, 1961.

R. V. Vohra. *Mechanism Design: A Linear Programming Approach*. Cambridge University Press, 2011.

A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd Foundations of Computer Science - FOCS*, 1982.