Large-scale hierarchical optimization for online advertising and wind farm planning

Konstantin Salomatin

CMU-LTI-13-008

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Yiming Yang, Chair
Jaime Carbonell
Tuomas Sandholm
Wei-Ying Ma, Microsoft Reseach Asia

Submitted in partial fulfillment of the requirements
or the degree of Doctor of Philosophy
In Language and Information Technologies

Keywords: online advertising, sponsored search, wind energy, layout optimization, optimization, game theory

To my parents.

Abstract

This thesis develops a framework to investigate and design novel optimization methods for two important problems: computational advertising (particularly, sponsored search) and wind farm turbine-layout planning. Whereas very different in specifics, both problems share some common abstractions. The existing solution in sponsored search is based on a greedy pay-per-click auction and is suitable only for advertisers seeking a direct response. It does not apply to advertisers who target certain numbers of clicks in a predefined time period. To address this new challenge, we introduce a unified optimization framework combining pay-per-click auctions and guaranteed delivery in sponsored search. Our new method maximizes the revenue of the search engine, targets a guaranteed number of ad clicks per campaign for advertisers willing to pay a premium, and enables keyword auctions for all others. Results combining revenue to the search engine and click rates for the advertisers show superior performance over strong baselines.

The proposed framework is based on linear programming with delayed column generation for computational tractability at scale. We design a game theoretic approach to optimize the strategy for individual advertisers, i.e. to optimize their choices between auctions and guaranteed delivery, and analyze the behavior of the new market formed by our framework. Specifically, we introduce a new method for computing the approximate Nash equilibrium where an exact computation would prove computationally intractable. We rely on approximations of complex utility functions, a combination of simulated annealing and integer linear programming as our principled approach. Wind farm layout optimization is the selection of optimal locations for placement of large wind turbines taking into account factors such as topographical features, prevalent but non-constant wind direction and turbine-wake interference. Existing approaches are deficient in their inability to consider long distance turbine interference, changing wind speed and direction and multiple types of wind turbines in optimization. The dissertation develops an optimization framework based on a scalable divided-and-conquer strategy that enables scalability to real-world wind farm scales taking into account the aforementioned complexities in the optimization process. Essentially the process optimizes in a hierarchical manner at different levels of granularity. This hierarchical decomposition approach to optimization is common to both search-advertisement and wind-farm layout challenges.

Acknowledgments

First and foremost, I would like to thank my advisor, Yiming Yang, for her continuous support over the course of this PhD and for keeping me motivated and organized. Her enthusiasm for my work and insightful ideas made this thesis possible. I would also like to thank Tie-Yan Liu for the great opportunity to work at Microsoft Research Asia. The substantial amount of work was done during my one-year-long MSRA internship and I consider him to be my second advisor. I am thankful to my thesis committee members for their time and expertise: Jaime Carbonell, Tuomas Sandholm and Wei-Ying Ma.

I would like to express my gratitude to my former CMU advisor, Eugene Fink. Without his excellent supervision I would not have been able to start my PhD program.

CMU colleagues and MSRA folks, thank you for your help, feedback on my research and meaningful discussions: Siddharth Gopal, Abhay Harpale, Andrew Hsi, Abhimanyu Lad, Bin Gao and Taifeng Wang. I am also grateful to Datong Chen, Qi He and Peiji Chen for my wonderful Yahoo! Labs experience and for introducing me to the fascinating field of online advertising.

All my Pittsburgh friends, without you I would not have been able to survive through graduate school: Anton Chechetka, Maxim Makatchev (CMU will remember our foosball games), Oleksii Mostovyi, Nikolay Simakov, Maxim Chistokletov, Mihail Pivtoraiko and Mellon Institute soccer team.

I am forever indebted to my parents for their patience in raising, mentoring, educating and supporting me. They played a tremendous role in my choice of the path of science. Without their encouragement I would not have decided to pursue a PhD.

Last but not least, Karina, thank you for being by my side through the last years of this journey.

Contents

List of Figures

List of Tables

Common mathematical notations

x, ξ	most small Latin or Greek letters denote scalar variables or parameters
f, g, ϕ, ψ	some small Latin or Greek letters denote functions
$\mathbf{x}, \boldsymbol{\xi}$	bold Latin or Greek letters denote vectors
$\mathbf{A}$	capital bold Latin letters denote matrices
i, j, k	integer variables used to index vector or matrix components
x_i	scalar component of vector $\mathbf{x}$
$\mathbf{a}_i$	row or column (depends on the context) of matrix $\mathbf{A}$
a_{ij}	element of matrix $\mathbf{A}$
$\mathbf{I}$	identity matrix
$\mathbf{1}$	vector of ones, i.e. $\mathbf{1}^\intercal = \{1, 1, \dots 1\}$
p, P	used mostly for probabilities
∇	Nabla operator for full gradient
$\frac{\partial}{\partial x_i}$	partial derivative operator

Chapter 1

Introduction

This dissertation addresses two open problems in scalable optimization: 1) Search-engine advertisement optimizing for revenue and end-user click-through rates, combining guaranteed delivery and keyword auctions, and 2) Layout of different types of turbines in a large-scale wind farm, taking into account wind speed, direction, and both short-range and long-range wake interference. Whereas the two problems are very different in their specifics, both engender greater complexity than simpler optimization problems addressed before in each domain, both present challenges in computational tractability, and both are addressed via a new hierarchical divide-and-conquer technique, in addition to domain specific approximate methods.

The first optimization problem arises in online advertising, specifically in sponsored search. The marketing strategies of online advertisers are of two kinds: those directly engaging with search engine responses and a pay-per-click auction strategy, and those focusing on guaranteed delivery (GD) of the number of total clicks and a pay-per-campaign strategy. It has been a common assumption that the advertisers in sponsored search target the direct response and prefer keyword auctions but in reality what they really care about is the number of leads (clicks) to the websites in a certain period of time. The keyword auction algorithms are greedy, they do not take into account global constraints and cannot guarantee the long term performance. The existing optimization algorithms that could take long term goals into account face serious scalability issues. Their other limitation is a lack of principled way to solve the problem at the right granularity level. Modeling the granularity of the allocation is essential to balance tractability and accuracy of the solution.

To solve this problem and satisfy the needs of the advertisers we introduce the optimization framework for joint modeling of keyword auctions and guaranteed delivery. In our framework the advertiser can choose one of the two investment strategies (auctions or GD) for the advertisement campaign. Optimal ad allocation is decided by a linear program with the combination of revenue and click-through rate as the objective function, and budget limits of auction advertisers and click

goals of GD advertisers as constraints. To model the problem at the right granularity level we cluster the query submissions according to predicted user intentions. To address the scalability we develop a new hierarchical decomposition technique that partitions the problem into multiple independent sub-problems whilst the parameters of the partitioning are decided by the master optimization task. Thus, the framework is reformulated into a bi-level program. This technique results in 1000-fold improvement of computation time. Overall, we achieve 3% improvement of search engine revenue and 13.2% improvement of click-through rate.

We investigate our framework not just from the search engine's perspective but also from the advertisers' perspective. Specifically, we optimize the choices of individual advertisers between auctions and GD in the competitive sponsored search environment using the instruments of game theory. Nash equilibrium is the central concept in game theory that allows the analysis of the strategic interaction of multiple players. However, no existing methods can compute the equilibrium of a game with thousands of players whose utility functions cannot be expressed as closed-form expressions. To address the above difficulty, we propose a novel approach to finding Nash equilibrium via local linear approximations of the advertisers' utility functions. We use a combination of simulated annealing and integer linear programming to find the ϵ-approximate equilibrium for a set of linear utilities. We find the solution with 94.5% advertisers happy with their strategy, compared to 78.5% achieved by the baseline. The potential aggregate loss of unhappy advertisers is only 0.5% of the social welfare compared to 8% of that by the baseline.

The second problem investigated in this work is the wind farm layout optimization. Little work has been done on constructing global optimization frameworks for wind farm layouts. A lot of decisions in the industry are made in an adhoc manner which leads to poor and inefficient designs even for small 1-turbine farms. Such important factors as multiple types of turbines in a single layout, varying wind speed and direction and long distance turbine interference (wake effect) are often ignored in the scientific literature due to a lack of scalable optimization algorithm.

To address the limitations of the existing solutions we propose an integer linear programming formulation of wind farm optimization on a coordinate grid. We model all the factors mentioned above, namely, multiple types of turbines in a single layout, varying wind speed and direction and long distance turbine interference. The main challenge is the efficiency, determined by the size of the problem and the computational complexity of the solution. Our solution is based on hierarchical decomposition. We partition the wind farm into a number of small regions. The optimal placement of a given number of turbines within a region is solved independently of other regions. To find the optimal distribution of turbines across the regions we formulate the master optimization problem. This decomposition allows us to use a superior quadratic model of turbine interference in the master problem compared to the linear model used in the baseline solution. For large (10km by 10km) wind farms with hundreds of turbines our solution achieves up to 25% improvement of power output and several orders of magnitude speedup.

The thesis is organized as follows. We describe our solution for sponsored search in chapters

2, 3 and 4. Each chapter addresses different limitations of the existing solutions and explores different aspects of the problem. In chapter 2 we introduce the field of online advertising, formulate our joint modeling framework for auctions and GD in sponsored search and derive our optimization algorithm. Chapter 3 concerns two interrelated issues: granularity of the allocation and scalability of the algorithm. We introduce our hierarchical optimization solution. In Chapter 4 we study the optimal choice between auctions and guaranteed delivery and present our algorithm for Nash equilibrium computation. In chapter 5 we describe our wind farm layout optimization solution.

Chapter 2

Unified Optimization Framework for Auctions and Guaranteed Delivery in Sponsored Search

2.1 Introduction

Sponsored search is an important means of Internet monetization, and is the driving force of major search engines today. It has sustained a market of tens of billions of dollars, and the market size is still growing very fast [35] [72]. The success of sponsored search is in part due to its business model, and in part due to its strong technical foundation in information retrieval, data mining, machine learning, and algorithmic economics.

Currently, sponsored search works in the following manner. When a user submits a query to a search engine, in addition to the regular search results (also called organic), a list of the advertisements (ads) is also presented to the user. The ads can be displayed in the side panel, on top of the organic results or even mixed with them, see the figure 2.1. If the user clicks on one of these ads, the advertiser will be charged with a certain amount of money. The selection, ranking, and pricing of the ads are determined by an auction mechanism which is usually referred to as a pay-per-click auction, and the most popular mechanism is generalized second price (GSP) [26], where the ads are ranked according to their expected values (the predicted click probability of an ad times the bid price given by the advertiser), and the advertisers are charged according to the second-price rule. That is, the cost per click is the minimum bid price required to keep a given ad in its current rank position. An increasing amount of research has been conducted on improving the pay-per-click auction, such as prediction of the click probability in a more accurate manner [16] [32] [73], optimization of ad presentation to increase click probability [41], and introduction of more effective ranking and pricing rules [26] [25] [22] [58] [15] [64].

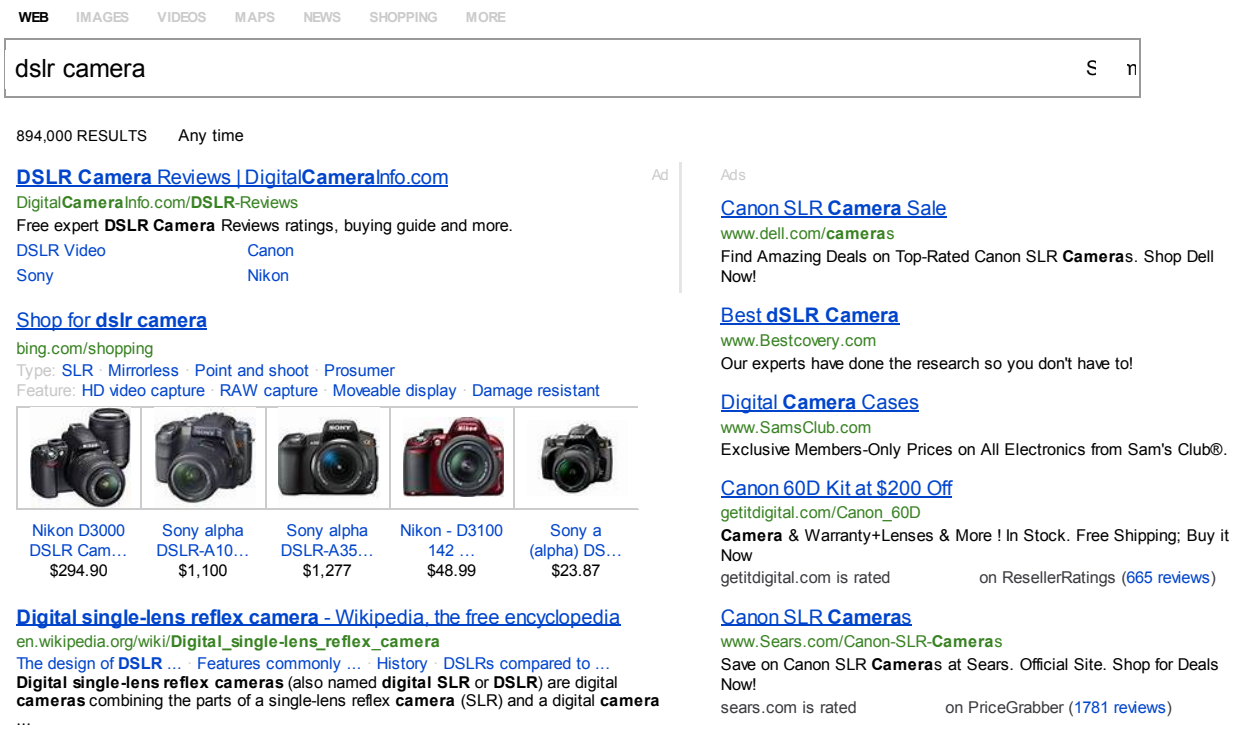


Figure 2.1: Screenshot of a search results page. The search string is "dslr camera". The top result in the left panel and the whole right panel are paid ads.

Another popular marketing strategy in online advertising is called guaranteed delivery (GD) where the ads are grouped by theme into campaigns, and each campaign has a price for the pre-specified number of clicks or impressions, called deliveries. Each campaign is typically involved with multiple ads, and a pre-specified period of time for each ad. Comparing GD with the pay-per-click auction, the latter focuses on the winning of each individual auction (i.e., a micro-level optimization), while the former concentrates on the campaign-level (macro-level) success for grouped ad impressions or clicks. Most popular GD solutions are based on offline optimization algorithms, adjusted for online setup [39, 67, 68]. Linear Programming (LP) is the core of these techniques due to its flexibility and the availability of efficient algorithms. The limitations of LP include inability to model all-or-nothing campaigns when the advertisers only pays if the click goal is reached or bonus campaigns when the price per click increases if a certain click goal is reached [9]. Presence of such campaigns requires integer linear programming that is much harder to solve.

Can the strengths of both pay-per click auction and GD be combined in sponsored search? Furthermore, how to model both in a unified optimization framework? Those meaningful ques-

tions have not been studied before. Answering these questions with a principled solution is one of the contributions of this work and this chapter specifically. By doing so, we will be able to provide the advertisers with more powerful and flexible means to achieve their marketing goals, as well as help search engines to attract more advertisers in a long run.

The pay-per-click auction has been the main focus in sponsored search while GD is a dominating solution in display advertising. Since GD has not been used in sponsored search before we explain our interest in this model and why it is important to introduce it to the sponsored search. We start with the intuition and back it up with a simple study of the advertisers in Microsoft's Bing search engine (www.bing.com). Obviously both strategies are important for search engine revenue and for the needs of the users (advertisers and searchers). Even the advertisers, who target immediate response, measure their success at the campaign level, e.g., the total number of ad impressions, clicks and actions, and the average cost per click in a certain period of time. For instance, these metrics are included in the report generated by Microsoft Ad Center for the advertisers. Other advertisers, given the high user traffic in search, want to promote their brands through sponsored search, in addition to the conventional channels such as TV and newspaper for brand marketing [55]. Such advertisers also set campaign-level budgets for pre-defined numbers of reaches to the users (i.e., ad impressions) or the numbers of leads (i.e. clicks) to the advertisers' websites. There are many technical articles discussing how to enhance brand awareness by the means of sponsored search [44] [71]. Some of those articles show that brand-oriented ads for news queries or other indirect/irrelevant queries may hurt the experiences of search users, and suggest that the advertisers should bid on relevant keywords even if their campaign goals are brand awareness.

Why cannot the current approaches in pay-per-click auctions provide guaranteed campaign level performance? The reason is that the auctions are run greedily per individual query, not taking into account the total budgets and the total number of targeted deliveries (ad impressions or clicks) as global optimization criteria. As a result, often a significant portion of the budgets allocated by advertisers remains unused, meaning that the advertisers cannot reach their expected campaign goals. Our analysis of one-month of auctions of Bing search engine shows that the used-budget ratio is only around 10% on average, see the figure 2.2. This problem is caused by the inexpressiveness of auctions and is acknowledged by other researchers. Attempts to incorporate global constraints into sponsored search auctions inevitably lead to the need of global optimization: [1, 23, 56]. Existing approaches mainly focus on the advertisers whose budgets are running out, and not on the advertisers who cannot reach their targets in spending and click-through numbers. Feldman et al. mention in [30] that their model can accommodate both budgeted and inventory constraint bidders, however, no actual investigation was done. Other limitations of the existing models include (not necessarily simultaneously) single slot auctions instead of multiple slot auctions, the first price payment rule instead of the second price payment rule, payment per impression instead of payment per click and the lack of experimental results.

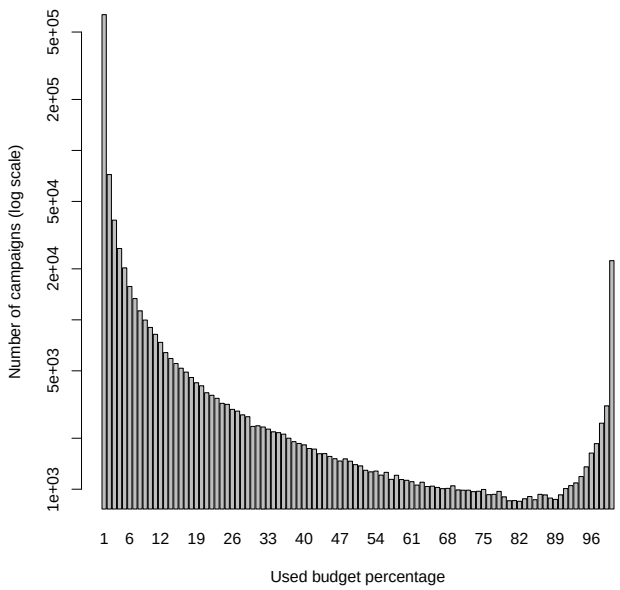


Figure 2.2: Distribution of campaigns according to the used budget ratio in one-month auctions of Bing search engine. The campaigns are put into 100 bins that correspond to 1-100 percentiles. The y-axis (in log scale) measures the number of campaigns in the corresponding bin.

The straightforward adaptation of the existing GD mechanisms (with efficient optimization solutions) in sponsored search might not work well. GD in display advertising provides the premium advertisers with a performance guarantee (or in other words a quality of service), e.g., a pre-defined number of impressions of their ads. The advertisers usually need to pay more for the guaranteed ad impressions than the non-guaranteed ad impressions. In contrast, small-business advertisers play an important role in sponsored search, and both guaranteed and non-guaranteed (auction-based) impressions and/or clicks are important because each kind is desired by a significant portion of the advertisers (premium advertisers vs. small-business advertisers).

To combine the strengths and to address the limitations of the two models, we introduce a unified optimization framework for combining GD and pay-per-click auctions. Specifically, we formulate the optimization problem using linear programming. The objective is the revenue of the search engine or alternatively a combination of the revenue and the number of clicks. The constraints include the budget limits in auctions and the guaranteed click numbers in GD. Inspired by [1], we use delayed variable generation also known as delayed column generation [46] to handle the exponential number of combinatorial constraints in multiple item auctions. To se-

lect the best variables (each variable corresponds to a ranked list of ads), we solve a dynamic programming problem that takes the special structure of the ad allocation and pricing mechanisms into account. To our knowledge, this is the first work in the literature of sponsored search that addresses advertisers' needs of both direct-response marketing and brand marketing using joint optimization of GD and auctions.

The chapter is organized as follows. First, we introduce the existing keyword auctions and guaranteed delivery solutions. In section 2.4 we introduce the unified framework: we formulate the problem as a linear program with delayed column generation. Section 2.5 presents some natural modifications of the formulation. In section 2.6 we explain how to apply the results of the offline optimization to the online allocation. In section 2.7 we report our experimental results and the comparison with the representative baseline methods.

2.2 Generalized Second Price Keyword Auction

A keyword auction based on Generalized Second Price (GSP) auction is commonly used by major search engines in sponsored search [26]. The advertiser selects a set of search keywords that are relevant to his ad. For example, a sports goods store willing to advertise its new summer collection can select the keywords "shoes", "running" and "Nike". The advertiser also indicates the bid: how much he is willing to pay if his ad is clicked and the query contains one of the selected keywords. For example, 10 cents for a query that contains the keyword "shoes" and 50 cents for a query that contains the keyword "running" or "Nike". Whenever a new query is submitted the search engine identifies all the advertisements that bid on one or several query terms; ranks their ads and displays several top-ranked ads alongside the search results. The advertiser pays the search engine if his/her ad is clicked (a so called cost-per-click model, CPC).

Several ads of one advertiser that share a single idea and theme are grouped into advertising campaigns. The advertiser may select different keywords and bids for different ads of one campaign. For example "The new collection of basketball shoes" advertisement can bid on basketball-related keywords while "The new collection of running shoes" can bid on running-related keywords. It is important to note that the advertisers set specific goals, such as budget limitations, for campaigns and not for individual ads.

For every query the search engine selects relevant ads from the campaigns whose remaining budget exceeds a certain threshold and ranks them according to their quality and the bid amount. Usually the probability of a click is used as a proxy for the ad quality. The click probability is often computed as a product of two independent quantities: the advertisement specific click probability and the position bias that depends on the placement of the ad:

$$P_{click} = P_{ad}(query, user, ad) \cdot P_{bias}(pos) \tag{2.1}$$

The same ad has higher chance to be clicked when displayed on the top of the page rather than in the bottom. The position bias is normalized so that the best spot $pos = 1$ on top of the page has $P_{bias}(1) = 1$. The ad specific probability, $P_{ad}(query, user, ad)$, in this case corresponds to click probability when the ad is displayed on the top spot. It depends on the relevance of the ad to the $query$ and the needs of a particular $user$ who submits the query. The problem of click probability prediction is out of the scope of this work. The reader can get more information on this topic in [16, 32, 41, 73]. In this work we use pclick predictions generated by a production regression module of the Bing search engine. We will use the term *pclick* as the abbreviation for "click probability of an ad".

The search engine ranks the ads according to the rank score which is a product of the bid amount and ad-specific click probability P_{ad}:

$$Score = Bid(query, ad) \cdot P_{ad}(query, user, ad) \tag{2.2}$$

The ads above the quality threshold will be displayed. There is a limit on the maximum number of ads that can be displayed for a particular query. The ad with the highest score will be displayed on the best spot with the highest P_{bias} (usually the top spot[1][2].), the ad with the second highest score will be displayed on the second spot etc. The amount that the advertiser pays when the ad is clicked is determined by the generalized second price rule and equals the smallest bid that would preserve his position in the ranking:

$$Bid(a_i) \cdot P_{ad}(a_i) \geq Bid(a_{i+1}) \cdot P_{ad}(a_{i+1}) \tag{2.3}$$

In this formula the argument a_i is the advertisement displayed in the position i. The arguments $query$ and $user$ are omitted for the sake of compactness. Accordingly, the smallest bid amount $Bid^{min}(a_i)$ for the ad a_i to preserve the position above the ad a_{i+1} is

$$Bid^{min}(a_i) = P_{ad}(a_{i+1})/P_{ad}(a_i) \cdot Bid(a_{i+1}) \tag{2.4}$$

It is easy to see that the expected payment for the ad in position i is:

$$E[Payment(a_i)] = P_{click}(a_i)Payment(a_i) = P_{ad}(a_{i+1})Bid(a_{i+1})P_{bias}(i) \tag{2.5}$$

and the expected payment for the whole list is:

$$E[Payment] = \sum_{i=1}^{K} P_{ad}(a_{i+1})Bid(a_{i+1})P_{bias}(i) \tag{2.6}$$

[1]The spots above the organic search results have the highest quality, then go the spots in the side panel. The positions above the main search results may have their own quality threshold that is higher than the threshold for the side-panel. In this work we assume that the threshold is the same.

[2]Sometimes the advertisers prefer lower positions because the ads displayed in these positions have higher chance of conversion give the click. The conversion is the action that the user performs on the landing page of the ad, e.g. purchase, subscription etc [7]. Modeling such preferences is out of the scope of this work.

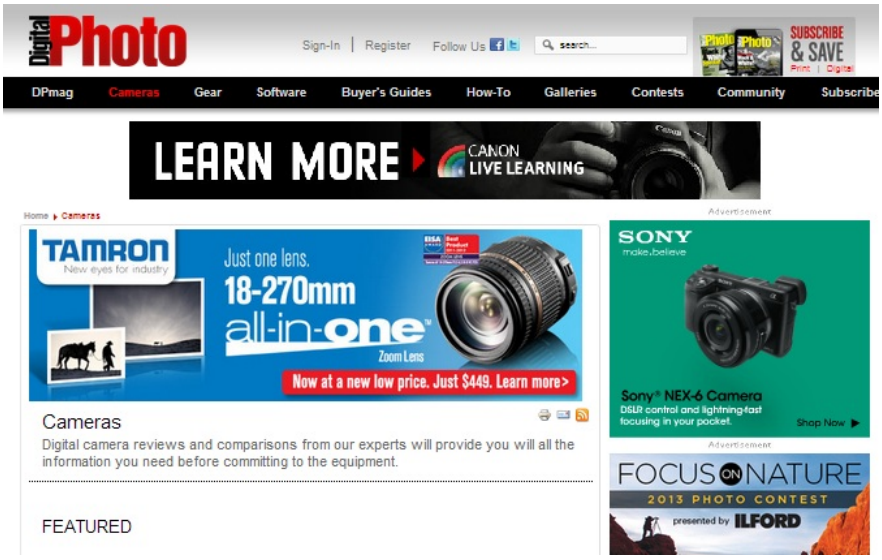


Figure 2.3: Screenshot of a www.dpmag.com. There are two display ad banners on the right.

The payment for the K-th, last displayed, ad can be slightly different from this formula. If the quality of the ad ranked at position $K+1$ is below quality threshold, then the payment for the ad in position K is either a reserved price [54] or depends on the threshold, rather than on the ad in position $K+1$:

$$Bid^{min}(a_K) = QualityThreshold/P_{ad}(a_K) \tag{2.7}$$

The advantage of the keyword auction is that it is a fast online algorithm. It jointly considers all the ad positions on a page therefore it is possible to impose local constraints on the list (e.g. do not put the ad of the same advertiser twice). However it runs in a greedy manner per individual query, not taking into account the total budgets and total number of targeted deliveries (ad impressions or clicks) as global optimization criteria.

2.3 Guaranteed Delivery in Display Advertising

Display advertising is another type of online advertising. It differs from sponsored search in many aspects, but we want to borrow some of the ideas and techniques used in display advertising and apply them to sponsored search.

In display advertising the advertisements are placed on predefined banner spots of the web-pages. These ads usually have rich graphical representation, see figure 2.3. Sometimes the ads may contain audio, video or interactive flash environment. In contrast, the body of a sponsored

search ad is just a short text message with a URL. In display advertising brand awareness is often the main goal of the advertiser. Such advertisers pay per campaign. An example of the advertiser's campaign goal can be "display 100K ads to young females in California". This marketing strategy is called guaranteed delivery (GD). The target of a GD algorithm is to satisfy the number of ad impressions requested by the advertisers. Inevitably GD algorithms are based on offline optimization or manual (greedy) approaches. Whenever a user visits a web page there is an opportunity to display one or several advertisements. The union of all user visits ($\langle user, page \rangle$ tuples) is called inventory. The inventory is grouped by the combination of feature values (i.e. age, location, page etc.). The goal of the ad allocation mechanism is to match this inventory (supply) with advertisers' goals (demand). This is a standard bi-partite supply-demand matching problem (Figure 2.4) that can be solved using Linear Programming (LP)[3], [19]:

$$\begin{aligned} & \max_x f(x) \\ \text{s.t.} \quad & \sum_{i:(i,j)\in E} x_{ij} \geq m_j \quad \forall j \\ & \sum_{j:(i,j)\in E} x_{ij} \leq n_i \quad \forall i \\ & x_{ij} \geq 0 \end{aligned} \tag{2.8}$$

Here $f(x)$ is a convex (often linear) objective function that characterizes quality of the assignment, x_{ij} is the number of times we want to show ad j to the users group i. The set of edges in the bipartite graph, E, encodes the matches between advertisers targeting preferences and the users group (Figure 2.4). The first type of constraint is the demand constraint, it guarantees that the ad j will be displayed at least m_j times. The second type of constraint is the supply constraint, it guarantees that the number of ads displayed to users group i does not exceed the number of visits. The optimization is performed offline in a batch mode. The number of future visits (left-hand side of the graph) needs to be predicted [3].

The advantage of the formulation (2.8) is the ability to satisfy global campaign goals. However this solution does not take into account the needs of the advertisers interested in direct response and who wish to adhere to an auction in sponsored search. Particularly it is not possible to model local constraints (such as GSP pricing rule) between the ads displayed on the same page.

Spot auctions are also used in display advertising mostly as supplementary mechanism. For example, to allocate ads for extra inventory that results from inaccurate forecast. Usually the cost

[3]Note, that this LP solves the dispatch problem: how many ads to display for existing campaigns already accepted by the publisher. It cannot solve the campaign acceptance problem, i.e. making a Yes/No decision whether to accept the campaign proposed by the advertiser or to reject it.

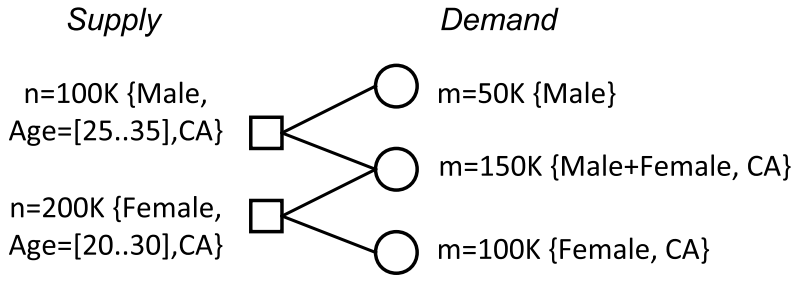


Figure 2.4: An example of a bipartite supply-demand graph in display advertising. Supply nodes on the left-hand side correspond to the number of user visits, demand nodes on the right-hand side correspond to the impression goals of the advertisers.

per impression sold in spot auctions is much lower than the cost per impression of a guaranteed delivery campaign. Given the price difference these two mechanisms do not compete directly.

2.4 Unified Optimization Framework

In this section we describe the approach that combines together keyword auctions (section 2.2) and guaranteed delivery (section 2.3) in a unified optimization framework. To the best of our knowledge this is the first time when these two delivery mechanisms are combined together in an optimization formulation in sponsored search (there exist solutions for display advertising, e.g. [68], that are not directly applicable to our domain). We start with the problem formulation in 2.4.1 and describe technical details of the solution in the subsequent sections 2.4.2 and 2.4.3.

2.4.1 Problem Formulation

In our proposed mechanism we allow two types of advertisers. Some advertisers are interested in auctions and bidding on relevant keywords, we call them auction-type advertisers and their campaigns accordingly - auction campaigns. The others prefer to specify the number of clicks as the explicit goal of the campaign, they select relevant keywords for their ads, we call them GD advertisers and their campaigns - GD campaigns[4]. Whenever a query is submitted a ranked list of ads is displayed. This ranked list can contain both GD and auction ads. GD ads can take any positions and auction ads are partially ranked by the rank score (2.2) (product of predicted click probability and the bid). GD advertisers are charged per campaign, the penalty is imposed on the search engine if the advertiser's goal (number of clicks) is not met. In scientific literature

[4]Actually, one advertiser can run several campaigns: some of them can be auction-type campaigns and some of them can be GD-type campaigns. This situation can be addressed by our solution.

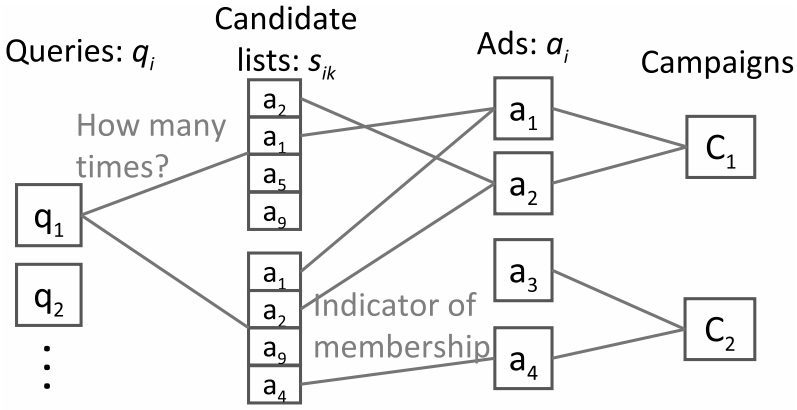


Figure 2.5: Supply-demand graph that represents our formulation. Supply nodes on the left-hand side correspond to query submissions. Demand nodes on the right-hand side correspond to campaign objectives. The decision variables determine how often to display a particular candidate ranked list of ads in response to a query.

this penalty is usually taken to be linear. The auction advertisers are charged per click according to the modified GSP pricing rule: the smallest bid that would preserve his position in the *partial* ranking of auction ads. Displaying the list of ads contributes to the goals of all the campaigns that have ads in it, this diagram is presented on figure 2.5.

Let us assume for now that we can enumerate all possible ranked lists of ads that are consistent with the ranking criteria above and are not longer than the maximum number of ad positions on one page. We will elaborate on this assumption in the end of this section. How often do we need to display a particular ranked list of ads in order to satisfy the goals of the advertisers, the search engine and the users? To answer this question we formulate and solve the following Linear Program (see the notation in Table 2.1):

$$\max_{x_{ik},\xi_c} \sum_i \sum_k \alpha_{ik} x_{ik} + \sum_{c \in GD} (d_c - \mu_c \xi_c) \tag{2.9}$$

$$\text{s.t.} \quad \sum_i \sum_k \sum_{j:a_j \in c} \delta_{ikj} x_{ik} \leq d_c \quad \forall c \in A$$

$$\sum_i \sum_k \sum_{j:a_j \in c} \beta_{ikj} x_{ik} + \xi_c \geq m_c \quad \forall c \in GD$$

$$\sum_k x_{ik} \leq 1 \quad \forall i$$

$$x_{ik}, \xi_c \geq 0$$

Table 2.1: Problem notation

Variable	Description
$q_i \in Q$	an individual query and the set of all queries
n_i	number of times we expect q_i to be issued
S_i	the set of candidate ranked lists of ads that can be displayed in response to a query q_i
$s_{ik} \in S_i$	a ranked list of ads that can be displayed in response to a query q_i
$x_{ik} \in [0, 1]$	the frequency of displaying the list s_{ik}
$a_j \in c$	an individual advertisement that is a part of a campaign c
$c(j)$	the campaign that the ad a_j belongs to, i.e. $c(j) = c : a_j \in c$
A	the set of auction type campaigns, for example $c \in A$
GD	the set of GD type campaigns, for example $c \in GD$
d_c	a budget of a campaign c
m_c	a click requirement for a GD campaign
μ_c	a penalty that the search engine pays to a GD advertisers for 1 underdelivered click
ξ_c	the number of underdelivered clicks for a GD campaign $c \in GD$
z_{ikj}	an indicator variable, $z_{ikj} = 1$ if an ad a_j is a member of a list s_{ik}
p_{ikj}	the payment if an ad a_j in the list s_{ik} is clicked; $p_{ikj} = 0$ for GD ads. For the auction ads the payment is determined by the updated GSP pricing rule
c_{ikj}	the click-through rate (CTR) for an ad a_j in the list s_{ik}

We use i to index queries, j to index ads and ik to index candidate ranked lists relevant to query q_i. The decision variable of the problem x_{ik} is the frequency of displaying the list s_{ik} in response to a query q_i. The objective function is the revenue of the search engine. The first summation of the objective function is the expected revenue from the auction campaigns. The coefficient α_{ik} is the expected revenue for n_i displays of the list s_{ik} (the summation over member ads). Only auction ads contribute to this coefficient:

$$\alpha_{ik} = n_i \sum_{j:c(j)\in A} c_{ikj} p_{ikj} z_{ikj} \tag{2.10}$$

The second summation of the objective function is the revenue from GD campaigns penalized by underdelivery. We express underdelivery using slack variables ξ_c. The first constraint guarantees

that the expected spending of the auction campaign c is limited by the campaign budget d_c. The second constraint guarantees that the GD campaign c gets the required amount of clicks m_c. The coefficients of the constraints matrix δ_{ikj} and β_{ikj} represent the expected payment and the expected number of clicks for an ad a_j in the list s_{ik}.

$$\delta_{ikj} = n_i c_{ikj} p_{ikj} z_{ikj} \tag{2.11}$$

$$\beta_{ikj} = n_i c_{ikj} z_{ikj} \tag{2.12}$$

The formulation (2.9) explicitly satisfies the following global goals: 1) maximize the search engine revenue, 2) guarantee the click requirements of GD campaigns and 3) provide equal opportunities to auction campaigns. We also expect our solution to 4) improve user's satisfaction expressed by the average click-thru rate (CTR). Note, that unlike display advertisers who are interested in a guaranteed number of impressions, sponsored search advertisers are interested in a guaranteed number of clicks. This means that the user's satisfaction is an implicit objective.

We will also use two variants of vector notation to make (2.9) more compact. In the first variant (below) we collapse the summation over k (summation over query-specific candidate ranked lists) into vector products.

$$\begin{aligned} \max_{\mathbf{x}_i, \xi_c} & \sum_i \boldsymbol{\alpha}_i^\intercal \mathbf{x}_i + \sum_{c \in GD} (d_c - \mu_c \xi_c) \\ \text{s.t.} \quad & \sum_i \boldsymbol{\delta}_{ic}^\intercal \mathbf{x}_i \leq d_c \quad \forall c \in A \\ & \sum_i \boldsymbol{\beta}_{ic}^\intercal \mathbf{x}_i + \xi_c \geq m_c \quad \forall c \in GD \\ & \mathbf{1}^\intercal \mathbf{x}_i \leq 1 \quad \forall i \\ & \mathbf{x}_i, \xi_c \geq 0 \end{aligned} \tag{2.13}$$

where $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots\}$, $\boldsymbol{\alpha}_i = \{\alpha_{i1}, \alpha_{i2}, \dots\}$, $\boldsymbol{\delta}_{ic} = \{\sum_{j:a_j \in c} \delta_{i1j}, \sum_{j:a_j \in c} \delta_{i2j}, \dots\}$ and $\boldsymbol{\beta}_{ic} = \{\sum_{j:a_j \in c} \beta_{i1j}, \sum_{j:a_j \in c} \beta_{i2j}, \dots\}$. In the second variant we additionally collapse the summation over i (summation over queries) into vector products:

$$\begin{aligned} \max_{\mathbf{x}, \xi_c} & \boldsymbol{\alpha}^\intercal \mathbf{x} + \sum_{c \in GD} (d_c - \mu_c \xi_c) \\ \text{s.t.} \quad & \boldsymbol{\delta}_c^\intercal \mathbf{x} \leq d_c \quad \forall c \in A \\ & \boldsymbol{\beta}_c^\intercal \mathbf{x} + \xi_c \geq m_c \quad \forall c \in GD \\ & \mathbf{1}^\intercal \mathbf{x}_i \leq 1 \quad \forall i \\ & \mathbf{x}, \xi_c \geq 0 \end{aligned} \tag{2.14}$$

where vector $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$ is the result of concatenation of the vectors $\mathbf{x}_i$. Correspondingly, $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$ and $\boldsymbol{\beta}_c$ are the results of concatenation of $\boldsymbol{\alpha}_i$, $\boldsymbol{\delta}_i$ and $\boldsymbol{\beta}_{ic}$.

Let us return to the assumption that we can enumerate all candidate ranked lists of ads, also called *slates*, and justify our choice of a variable in the problem. Each variable x_{ij} corresponds to one slate, a ranked list of ads that can be displayed in response to a query q_i. A slate is an atomic element in the optimization problem above, it acts as a whole. The sufficient parameters that describe each slate are its expected revenue and its expected contribution to the campaign specific constraints (equations (2.10), (2.11) and (2.12)). On the other hand each slate is in turn a complex object that consists of multiple ads. It is impossible to use an individual advertisement as an atom of the optimization because of the dependencies among the ads in a slate: the auction ads must follow ranking constraints and GSP pricing constraints. Even more ad dependencies can be potentially added in the future. For example, one area of active research in online advertising is the externalities effect when each ad affects the pclicks of its neighbors. We make our framework flexible to accommodate the prospective externality models.

The use of slates as atoms of the optimization was proposed in [1]. The downside of this approach is a large number of feasible ranked lists of ads that leads to a large number of variables in the linear program. The remedy to this problem is the fact that 1) only few "good" candidate lists are worth considering; and 2) these "good" candidate lists can be efficiently generated on the fly. To identify these good candidates we formulate a series of auxiliary optimization problems in section 2.4.2. To solve these problems we exploit the special structure of (2.9) in section 2.4.3.

2.4.2 Delayed Column Generation

Linear programs with a huge number of variables and a small number of constraints can be solved using delayed column generation technique [46]. The number of non-zero (active) variables in the solution is small and to find them it is sometimes not necessary to consider all the variables. Indeed, the number of non-zero variables of a solution, located in the vertex of a polyhedron (feasibility set of LP), is restricted by the number of constraints: $|A| + |GD| + |Q|$. Delayed column generation finds the active variables iteratively. It starts with an LP with a small subset of variables, solves this LP (also called *restricted* LP), finds the variable that is not in the subset and can improve the objective and adds it to the basis of the restricted LP. This procedure is similar to the iteration of the simplex method [17]. Delayed column generation and the simplex method only differ in the way they find a variable that enters the basis. The simplex method iterates through all the variables explicitly to find the best candidate. Column generation solves a complementary optimization problem to find the best candidate. Column generation is useful when the number of variables is very large and the problem structure can be utilized.

To increase the objective the new variable must have a positive reduced cost, the quantity

defined below. For a linear program in standard form:

$$\begin{aligned} \max_{\mathbf{x}} \mathbf{c}^\intercal \mathbf{x} \\ \text{s.t.} \quad A\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{aligned} \tag{2.15}$$

The reduced cost vector is given by

$$\mathbf{c}' = \mathbf{c} - A^\intercal \mathbf{y} \tag{2.16}$$

where $\mathbf{y}$ is the dual solution. We select a variable with the highest reduced cost coefficient to enter the restricted primal:

$$j^* = \arg\max_j c_j - A_j^\intercal \mathbf{y} \tag{2.17}$$

For a reasonably small number of variables (2.17) can be solved by direct iteration over j (exactly what the simplex method does). This is infeasible for our problem and we explore the special structure of (2.17) to formulate a complementary optimization problem. Let $\mathbf{y}$ be the vector of variables dual to the first two constraints of (2.9) and $\boldsymbol{\gamma}$ be the vector of variables dual to the third constraint of (2.9). The dual to (2.9) is:

$$\begin{aligned} \min_{\mathbf{y},\boldsymbol{\gamma}} \sum_{c \in A} d_c y_c - \sum_{c \in GD} m_c y_c + \sum_i \gamma_i \\ \text{s.t.} \quad \sum_{j:c(j) \in A} \delta_{ikj} y_{c(j)} - \sum_{j:c(j) \in GD} \beta_{ikj} y_{c(j)} + \gamma_i \geq \alpha_{ik} \\ y_c \leq \mu_c \quad \forall c \in GD \\ \mathbf{y}, \boldsymbol{\gamma} \geq 0 \end{aligned} \tag{2.18}$$

The reduced cost of a candidate list s_{ik} is then given by:

$$cost(s_{ik}) = \alpha_{ik} - \sum_{j:c(j) \in A} \delta_{ikj} y_{c(j)} = \sum_{j:c(j) \in GD} \beta_{ikj} y_{c(j)} - \gamma_i \tag{2.19}$$

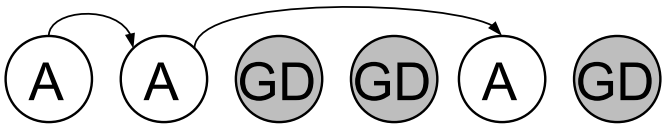


Figure 2.6: Dependencies in a ranked list of ads, that affect the objective.

To solve (2.17) we find the best candidate s^*_{ik} for each query q_i:

$$k^* = \arg\max_k cost(s_{ik}) = \arg\max_k \sum_{j:c(j)\in A} \delta_{ikj} - \tag{2.20}$$
$$\sum_{j:c(j)\in A} \delta_{ikj} y_{c(j)} + \sum_{j:c(j)\in GD} \beta_{ikj} y_{c(j)} - \gamma_i$$
$$= \arg\max_k \sum_{j:c(j)\in A} n_i c_{ikj} p_{ikj} z_{ikj} (1 - y_{c(j)})$$
$$+ \sum_{j:c(j)\in GD} n_i c_{ikj} z_{ikj} y_{c(j)} - \gamma_i$$

The first summation in the equation is a reweighed expected revenue of the list, where each auction ad member a_j is scaled by $1 - y_{c(j)}$. The second summation is a reweighed expected number of clicks of GD ads in the list, where each GD ad member a_j is scaled by $y_{c(j)}$. The last term only depends on the query.

Expression (2.20) is essentially a weighted sum over all advertisements that participate in the ranked list s_{ik}. To find the best list it is not necessary to inspect all the candidates, it is sufficient to construct the sequence of ads that maximizes (2.20). What makes this task challenging is the auction advertisements whose payments depend on the next auction advertisement in the list through the GSP pricing rule, equation (2.5). We describe the solution to this problem in the next section.

2.4.3 Dynamic Programming

In this section we provide the solution to (2.20). Optimization of the objectives defined over the chains of objects, when the contribution of a current element depends not only on the element itself, but also on the next element in the chain, can be done by dynamic programming (DP). However (2.20) is more challenging because of the presence of two types of objects: GD ads (unchained) and auction ads (chained). If an auction ad is followed by one or more GD ads than its contribution depends not on the next object in the chain, but on the object located further down the list, Figure 2.6. Let us rewrite (2.20) in simpler notation. Our goal is to find a sequence

of ads:

$$s = \{a_1, a_2, ...a_p\}$$

that maximizes

$$cost(s) = \sum_{\substack{j=1..n \\ a_j \in A}} P_{click}(a_j) Pay(a_j)(1 - y_{c(a_j)}) + \sum_{\substack{j=1..p \\ a_j \in GD}} P_{click}(a_j) y_{c(a_j)} - \gamma_i / n_i \tag{2.21}$$

where $Pay(a_j)$ is the payment of the auction ad a_j that depends on the next auction ad in the list. And the click probability P_{click} is a product of ad-specific probability and the position bias

$$P_{click}(a_j) = P_{ad}(a_j) P_{bias}(j) \tag{2.22}$$

Let us assume that we have assigned the first $r - 1$ positions in this sequence:

$$\tilde{s} = \{a_1, a_2, ...a_{r-1}\}$$

and reached some state, that is a function of our previous decisions:

$$T_r = State(a_1 ... a_{r-1})$$

Then our residual goal is to assign the ads to the positions $r \dots p$ given the state T_r. The sufficient condition of optimality and applicability of dynamic programming is Bellman's principle: the decisions at steps $r \dots p$ that constitute optimal solution depend only on the state that we reach after steps $1 \dots r-1$ [6]. In this case we can use backwards induction to solve this problem. The complexity of backwards induction algorithm is $O(M^2 L)$ where M is the number of possible states and L is the length of a chain. We want to find a state function with small M value that satisfies the Bellman's principle.

We claim that the allocation of future auction ads only depends on the last auction ad in the sequence $\tilde{s}$ and its position. GD ads in the list are ranked by $P_{ad}(a_j) y_{c(j)}$ and the allocation of future GD ads depends on how many GD ads were already allocated. This fact is quite intuitive. The GD ads are not chained, therefore it is optimal to always select the best available GD ad first - this is the ad with the highest $P_{ad}(a_j) y_{c(j)}$ score. The auction ads are chained by the ranking and pricing rules, the residual objective for the future auction ads only depends on the last auction ad in the chain. The state function is the following tuple:

$$T_r = \{last\ auction\ ad, its\ position, number\ of\ GD\ ads\} \tag{2.23}$$

The size of this state space is bounded by ZLL where Z is the number of relevant auction ads and L is the length of the chain. Here Z bounds the number of possible values of the first feature of the state function and L bounds the number of possible values of the second and the third

features of the state function. Usually L is a small integer (maximum number of ads displayed on a page). If we ignore the positional bias then we do not need to know the position of the last auction ad (the second feature of the state function) and the state space can be reduced to ZL.

To actually find the optimal sequence we use the necessary condition that is expressed by Bellman's equation [6]. This equation connects the residual objective at state T_r with immediate gain from selecting an ad for the slot r : $g(a_r|T_r)$ and the residual objective for the next state $T_r + a_r \rightarrow T_{r+1}$:

$$F(T_r) = \max_{a_r}\{g(a_r|T_r) + F(T_{r+1})\} \tag{2.24}$$

The immediate gain if $a_r \in GD$ is

$$g(a_r|T_r) = P_{click}(a_r)y_{c(a_r)} = P_{ad}(a_r)P_{bias}(r)y_{c(a_r)} \tag{2.25}$$

i.e., the expected number of clicks scaled by $y_{c(a_r)}$, the dual variable of the campaign clicks constraint. The immediate gain if $a_r \in A$ equals the amount the last auction advertiser in the assignment T_r pays according to GSP. Assume that the position of the last auction ad in the list T_r is m. If we assign an ad $a_r \in A$ to the slot r then the advertiser who got the position m for his/her ad a_m would pay the amount to maintain his/her position against a_r, according to (2.5):

$$Pay(a_m) = Bid(a_r)P_{ad}(a_r)/P_{ad}(a_m) \tag{2.26}$$

This gives us the immediate gain:

$$\begin{aligned} g(a_r|T_r) &= P_{click}(a_m)Pay(a_m)(1 - y_{c(a_m)}) \\ &= Bid(a_r)P_{ad}(a_r)P_{bias}(m)(1 - y_{c(a_m)}) \end{aligned} \tag{2.27}$$

Note that the ad specific click through rate and the bid correspond to the newly added advertisement in the position r and the dual variable y and the position bias corresponds to the ad in position m.

The Bellman's equation (2.24) can be solved with the backward induction algorithm [6].

2.5 Model Extensions

In section 2.4 we formulated the optimization with the revenue maximizing objective function. Some researchers argue that search engine revenue is not a good choice of objective in advertising because it may conflict with the objectives of the advertisers. Indeed, if the search engine gets more money then the advertisers spend more money. They may not be happy about it. It may be desirable for the advertiser to get more clicks as a result of higher spending. If the advertiser gets

the same number of clicks then the higher spending is definitely not desirable. To address this problem we use two solutions: 1) represent the advertisers in the objective function or 2) allow the advertisers to limit the average cost per click for their campaigns.

Combined objective. The approach is quite intuitive. We use linear combination of the search engine revenue and the total number of clicks as the objective function:

$$Objective = Revenue + \lambda Clicks \tag{2.28}$$

Such composition shifts optimization objective from the search engine to the advertisers and the users. User satisfaction is a critical long term goal even if some immediate revenue has to be sacrificed. The experimental results in section 2.7 demonstrate that this objective function allows to greatly improve users' experience at practically no cost. With minor modifications the algorithm that solves the original problem can solve the modified problem. The reduced cost that we maximize in the dynamic programming stage (2.21) is:

$$\begin{aligned} cost(s) = &\sum_{\substack{j=1..n \\ a_j \in A}} P_{click}(a_j) \left[Pay(a_j)(1 - y_{c(a_j)}) + \lambda \right] \\ &+ \sum_{\substack{j=1..p \\ a_j \in GD}} P_{click}(a_j)(y_{c(a_j)} + \lambda) - \gamma_i / n_i \end{aligned} \tag{2.29}$$

It contains extra terms compared to (2.21), but the structure of the expression is the same. The cost maximization can be done by the backwards induction algorithm. We only need to update the immediate gains for auction and GD ads:

$$g(a_r|T_r) = Bid(a_r)P_{ad}(a_r)P_{bias}(m)(1 - y_{c(a_m)}) + \lambda P_{ad}(a_r)P_{bias}(r) \quad \text{for } a_r \in A \tag{2.30}$$

$$g(a_r|T_r) = P_{ad}(a_r)P_{bias}(r)(y_{c(a_r)} + \lambda) \quad \text{for } a_r \in GD \tag{2.31}$$

Constrained cost per click. The other way to ensure that the optimization is not biased towards search engine's objective is to put a cap on the average cost per click (CPC). This cap can be campaign specific. For GD campaigns CPC is already a constant, negotiated by the advertiser and the search engine. For auction campaigns, the cost per click is only limited by the largest bid in the advertiser's portfolio. This limit can be too loose and does not reflect the advertiser's expectation. It is easy in our framework to put a cap on the average cost per click on a per-campaign basis by adding a new constraint:

$$\boldsymbol{\delta}_c^\intercal \mathbf{x} \leq \mu_c \boldsymbol{\beta}_c^\intercal \mathbf{x} \quad \forall c \in A \tag{2.32}$$

where μ_c is the limit for the average CPC for a campaign c[5], β and δ are expected clicks and expected payments for a campaign c in each slate. With minor modifications the algorithm that

[5]For GD campaigns μ_c is already defined as an underdelivery penalty per click.

solves the original problem can solve the modified problem. Only the complementary optimization problem in the column generation stage (2.21) has to be updated. Let $y'_{c(a_j)}$ be the dual variable for newly introduced constraints (2.32). The updated reduced cost expression is:

$$cost(s) = \sum_{\substack{j=1..n \\ a_j \in A}} P_{click}(a_j) \left[Pay(a_j)(1 - y_{c(a_j)} - y'_{c(a_j)}) + \mu_{c(a_j)} y'_{c(a_j)} \right] \tag{2.33}$$

$$+ \sum_{\substack{j=1..p \\ a_j \in GD}} P_{click}(a_j) y_{c(a_j)} - \gamma_i / n_i \tag{2.34}$$

The expression has the same structure as (2.21), even though it contains the extra terms. The cost maximization can be done by the backwards induction algorithm. We only need to update the immediate gain for the auction ads:

$$g(a_r|T_r) = Bid(a_r) P_{ad}(a_r) P_{bias}(m)(1 - y_{c(a_m)} - y'_{c(a_m)}) + \mu_{c(a_r)} P_{ad}(a_r) P_{bias}(r) \tag{2.35}$$

This expression, like its original version, depends on the previous auction ad and its position, therefore the state function (2.23) remains the same.

2.6 Online Allocation

In this section we explain how to use our optimization algorithm in the online scenario. The algorithm described in the previous section performs offline optimization based on historical or predicted counts of query submissions and historical click probability predictions. We make other approximations to make our framework scalable: remove tail queries, group query submissions together into clusters, exclude very small advertisers from optimization. An offline optimization algorithm finds the best allocation of ads for this imperfect approximation of future query stream. In the online scenario we have to deal with all the imperfections that result from this approximation:

- **Non-accurate query counts**. This problem is known as supply forecasting, i.e. the forecasting of the left hand side of our allocation chart. The accuracy of the forecast depends on the chosen granularity of the supply nodes. The forecasts are more accurate if the granularity is coarse. For instance, it is easier to forecast the number of future submissions of a query "Microsoft office" by all users than the number of submissions of the same query by the female college students located in Pennsylvania.
- **Non-accurate pclick predictions**. This problem is closely connected to the previous one and to the choice of granularity. All query submissions within a single supply node of the allocation chart have the same averaged click probabilities. The coarse granularity has high pclick estimation error due to higher variance within larger clusters of submissions.

- **Removal of tail queries**. This is a necessary step to make the offline optimization scalable. Users submit millions of unique query strings. According to the power law, most of the submissions are very rare, and a small number of most frequent submissions cover a significant portion of the stream. In the online scenario we need to display ads for all queries including the ones that were not considered in the optimization.
- **Removal of small advertisers**. There exist a huge number of very small advertisers. In our framework they are most likely auction type advertisers. These advertisers have to be removed from the offline optimization to make it scalable. In the online stage, however, we must not ignore such advertisers.

A straightforward approach to online allocation is to use the primal solution directly. The primal solution tells us how often to display a particular ranked list of ads in response to a query. This is a very restrictive plan that does not allow much flexibility in the online stage. In fact, this approach suffers from all the problems mentioned above. For example, this plan ignores personalized click predictions that become available online. Instead it sticks to average click probabilities that were used offline. It is very costly to throw away such valuable piece of information. Also, this plan does not know how to allocate ads of small advertisers that were removed from the offline optimization.

An alternative and more practical approach uses the dual solution. It is based on the following idea: dual solution can be used to approximately reconstruct the primal; yet, the dimensionality of the dual is much lower.[6] Low dimensionality makes it more robust to imperfections in the training data used offline. Multiple works, that investigate the problem of online resource allocation, rely on this dual framework: [13, 23, 30, 47, 67]. We will first sketch the basics of the dual framework and then explain how to use it with our approach.

To explain the idea we will use a very general form of ad allocation problem. We assume one displayed ad per query, one ad per advertiser and only budget constrained advertisers. The notation is mostly consistent with the previous sections. The primal and the dual forms of this allocation problem are:

Primal

$$\begin{aligned} \max_x & \sum_{i,j} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_i \delta_{ij} x_{ij} \le d_j & \forall j \\ & \sum_j x_{ij} \le 1 & \forall i \\ & x_{ij} \ge 0 & \forall i,j \end{aligned}$$

Dual

$$\begin{aligned} \min_{y,\gamma} & \sum_j d_j y_j + \sum_i \gamma_i \\ \text{s.t.} \quad & \delta_{ij} y_j + \gamma_i \ge c_{ij} & \forall i,j \\ & y_j, \gamma_i \ge 0 & \forall i,j \end{aligned} \tag{2.36}$$

[6]This statement is true for LPs typical for ad allocation problems. It does not hold for an arbitrary LP.

Here x_{ij} is a decision variable whether to display an ad a_j for query q_i, c_{ij} is the benefit for the search engine of displaying this ad and δ_{ij} is the cost to the advertiser.[7] The budget constraint of the advertiser is denoted by d_j. In the dual problem y_j is dual to the budget constraint of the advertiser j, essentially, this variable assigns a cost on using the money of the advertiser j. For example, if the advertiser has unlimited budget, this cost is zero, i.e. we can spend his or her money and not being afraid of running out of budget. The variable that is dual to the supply constraint (one constraint per query), γ_i, does not play essential role in the algorithm.

The dimensionality of the primal solution is the number of non-zero variables x_{ij}. Usually, we have at least one non-zero variable per query. The dual solution combines $\mathbf{y}$ and γ whose dimensionalities correspondingly are the number of advertisers and the number of queries. The key point is that it is enough to know $\mathbf{y}$ to reconstruct the primal solution with good accuracy. Usually the number of advertisers is much smaller than the number of queries. The reconstruction is done in the following way. For each query q_i we select an advertisement a_j that maximizes the following quantity (conditioned that it is not negative):

$$cost(a_j) = c_{ij} - \delta_{ij} y_j \tag{2.37}$$

If the maximum cost is negative then we do not display any advertisements: i.e. $x_{ij} = 0 \ \ \forall j$, otherwise we display the winner (assuming there are no ties). This reconstruction algorithm only requires the values of y_j-s, one variable per advertiser. The following proposition explains why this algorithm works:

Proposition 2.6.1 *If there are no ties between ads and the cost of the winner is not zero then this algorithm perfectly reconstructs the solution of the primal problem. This solution will be integral, i.e.* $x_{ij} \in \{0, 1\}$.

Proof This proposition is easy to verify using Karush-Kuhn-Tucker conditions [10, pp.241-244]. First of all,

$$c_{ij} - \delta_{ij} y_j - \gamma_i \leq 0 \tag{2.38}$$

is a dual constraint (and also a reduced cost of a variable). And

$$x_{ij}(c_{ij} - \delta_{ij} y_j - \gamma_i) = 0 \tag{2.39}$$

$$(\sum_j x_{ij} - 1)\gamma_i = 0 \tag{2.40}$$

are the complementary slackness conditions. According to the complementary slackness condition (2.39) the variable x_{ij} is non-zero only if the reduced cost is zero. The rest variables should

[7] In the simplest formulation of ad words problem $\delta_{ij} = c_{ij}$ i.e. the revenue of the search engine is equal to the spending of the advertisers.

have negative reduced cost. Since we have no ties, there is only one candidate for a non-zero variable for query q_i, call it $x_{ij^*} > 0$ - the maximizer of (2.37). If this maximum is greater than zero, then γ_i is strictly positive and according to the complementary slackness condition (2.40), $\sum_j x_{ij} = 1$. This leads to $x_{ij^*} = 1$. If the maximum of (2.37) is less than zero, then the reduced cost is also less than zero and then $x_{ij^*} = 0$. ∎

The conditions in the proposition (no ties and non-zero cost (2.37)) are needed to exclude non-integer solutions, i.e. $x_{ij} = 0.4$. Why is it essential to have no ties? If two ads a_1 and a_2 are tied for the first place for query q_i they both have non-zero chance to be displayed. The optimal objective can only be achieved if the ads are displayed with the correct probabilities (proportional to x_{i1} and x_{i2} of the primal solution). The dual solution only reveals that these two ads can be displayed, but the relative probabilities are unknown. Is "no ties" assumption realistic? In fact, ties are likely to occur. The likelihood depends on the granularity of the formulation. Recall that the number of non-zero variables in the vertex of the polyhedron is at most $|Q| + |A|$ - the number of queries plus the number of the advertisers. If $|Q| \gg |A|$ then there approximately is one non-zero variable per query. In this case the ties are less likely. The solution in [23] suggests that the tie resolution can be achieved by smoothing or micro randomizations of the problem parameters.

The algorithm requires the dual variables y_j. It is not possible to solve the dual problem exactly because of a huge number of constraints (one per query submission). All the works mentioned above essentially differ in the way they approximate the dual solution. One class of algorithms is purely online. They start with $y_j = 0 \ \ \forall j$ and update the variables after each query submission based on displayed ads. Other algorithms use historical data to learn y_j by solving offline optimization, our framework is the example of such offline solution. Usually, offline optimization is formulated on the subsample of historical data due to scalability issues. Finally, hybrid approaches use historical data to train initial values of y_j and then update them as necessary in the online stage. We find the hybrid approach to be the most promising because it allows us to extend our solution to unseen data, particularly, unseen advertisers. In the mixture of auction and GD, some of the advertisers are very small (mostly auction advertisers) and are not included in the optimization. We simply set the initial dual value for these advertisers to zero. Each time the query is submitted we find the ranked list of ads that maximizes (2.37). For our formulation (2.37) is equivalent to the complementary optimization (2.17) of column generation. Its dynamic programming solution is already described in the previous sections.

2.7 Results

In this section, we describe the experimental settings and results that verify the effectiveness of our proposed approach.

2.7.1 Dataset and Experimental Setup

Queries. The evaluation is performed on the log data from Microsoft's Bing search engine. To make the problem manageable we use only the head (most frequent) queries. This is a standard technique (for example [1]) motivated by the power law query frequency distribution: few head queries cover a large proportion of query submissions. A significant improvement that our technique achieves in allocating the ads for frequent queries will result in a significant improvement overall. We discuss how to generalize our approach to the tail queries in section 2.6. Furthermore, we remove the queries that have low monetization potential, i.e. the queries that display very few ads (less than 3). After these steps we end up with 1000 unique query strings and 1.5M query submissions.

Advertisements and Campaigns. We identify all the campaigns relevant to the selected queries and then remove the campaigns with click counts less than 12. Optimization of such campaigns is not useful and only increases the dimensionality of the problem. We discuss this issue in section 2.6. After filtering we end up with 2,807 unique campaigns and 37,258 unique ads. All the parameters required for optimization are logged by the search engine: predicted click probabilities, bid keywords, bid values, campaign budgets, actual spending and the number of received clicks. In the real system click probabilities are computed for each query submission. This means that these probabilities can differ for the same query submitted by two different users. The model described in this chapter does not distinguish different submissions of the same query therefore we average the click probabilities. This limitation of the algorithm is addressed in Chapter 3.

Our data comes from the keyword auction log of the search engine and only contains Auction campaigns. We have no control of which advertisers and how many of them will switch their campaigns to GD therefore in the experiments we select such campaigns randomly. We use the number of clicks that is expected according to the log as the clicks goal of the GD campaign. We do not impose extra payments (premiums) for using Guaranteed Delivery in the current experimental setup. Normally such payments would exist and would provide more opportunities for optimization and higher expected revenue for the search engine. In Chapter 4 we investigate the strategy selection (Auctions vs. GD) behaviors of individual advertisers from the game theoretic perspective.

To simulate GD campaigns we randomly select $r\%$ of auction campaigns and convert them to GD campaigns. The ratio r will be driven by the market demand and will not be controlled by

the search engine. In the simulation we investigate the performance of the algorithm for different values of r between 0 and 1:

$$\begin{aligned} r &\sim Uniform[0,1] \\ \forall i: \ GD_i &\sim Bernoulli(r) \end{aligned} \tag{2.41}$$

Batch optimization and online allocation. Optimization is performed in the batch mode and should use the forecasts for the number of query submissions n_i. The errors of forecasting n_i affect the performance in the online stage. However the forecast for the head queries that we select for optimization is usually accurate and the error is not considered in the experimental setup. We leave the detailed investigation when this is not the case for the future work. In the online stage we use the output of the optimization as the probabilistic allocation plan: x_{ik} - is the probability to display a ranked list of ads s_{ij} in response to a query q_i

Baseline 1. We mimic the greedy GSP allocation mechanism (section 2.2) currently employed in the search engine. This baseline does not consider GD at all. By comparing to this baseline, we can see the value of our proposed concept, i.e., considering both auction and GD in sponsored search.

The order in which the queries arrive is random. Since this order can affect the outcome of the allocation we repeat this procedure 10 times and report the average results. The click counts in this randomized setting are expected clicks (assuming that changing the order of queries does not change the number of clicks on average).

Baseline 2. For a more representative baseline we split the available resources (query submissions) proportionally between auction and GD campaigns. Then we run GD and auction allocation mechanisms independently. This baseline considers both auction and GD, but not in a unified framework. It mimics the approach applied in display advertising in which GD and Auctions coexist but do not compete for the same inventory. By comparing to this baseline, we can see the value of our proposed joint optimization framework.

The number of ads that we display for a query is the same for all the methods. We use soft delivery constraints and set the per click penalty μ_c equal to the average per click payment of the campaign. Using hard delivery constraints leads to the infeasible formulation. This means that the click requirements that we have selected are challenging enough. We stop the optimization routine when the relative objective improvement for two consecutive steps drops below $\epsilon = 10^{-5}$.

Regularization. We evaluate the regularized version of our algorithm, proposed in section 2.5. We use weighted combination of the revenue and the number of clicks as the objective function. We test different values of the regularization parameter λ: 0, 0.1, 1, 3, and 10.

2.7.2 Unified Optimization Performance

There are three parties in the ad allocation process: the search engine, the advertisers and the users. To evaluate our framework we use the following four metrics that cover the interests of all

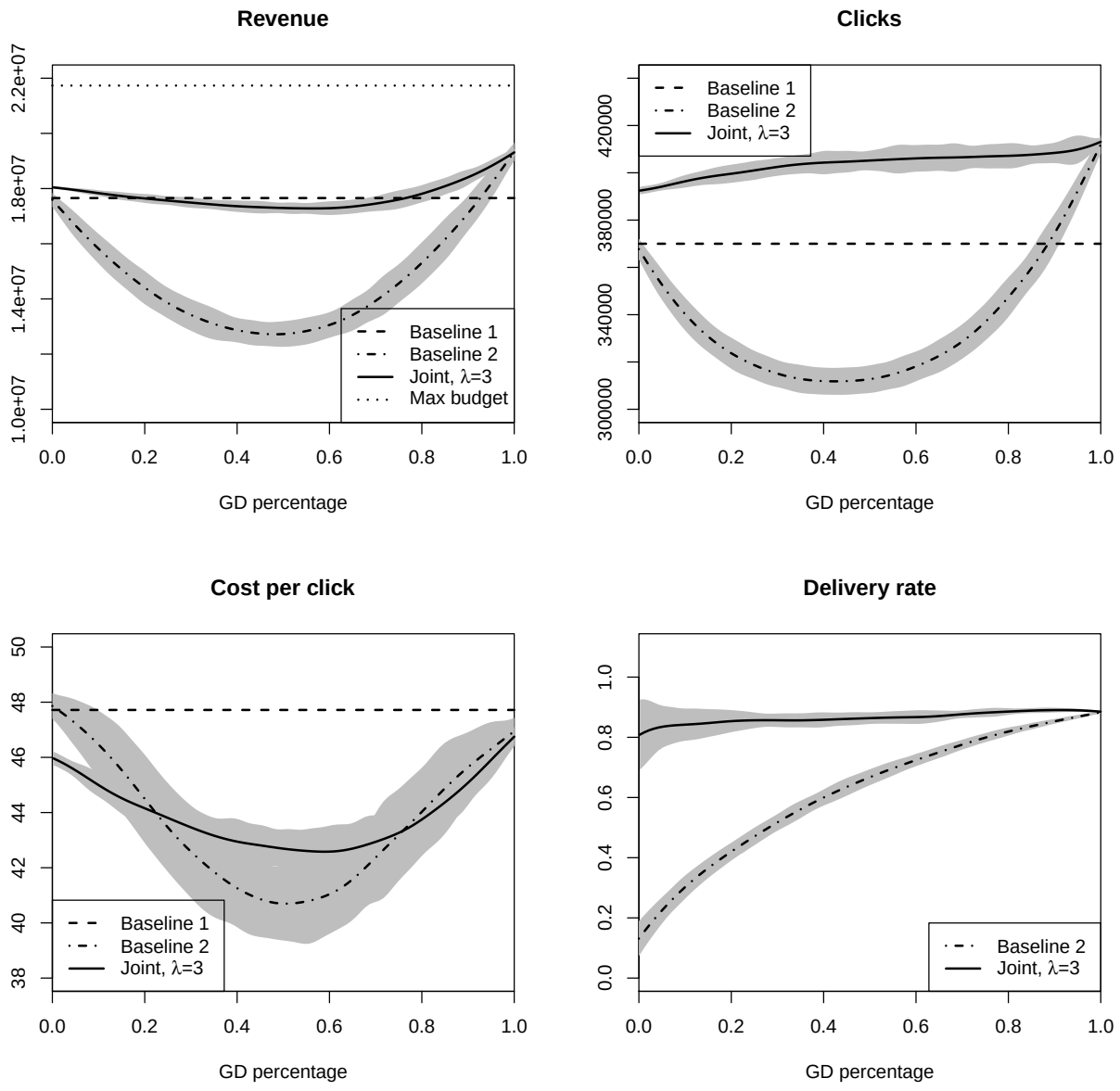


Figure 2.7: **Top Left**: Search engine revenue. **Top Right**: Expected number of clicks (also, proportional to average click trough rate). **Bottom Left**: Average cost per click. **Bottom Right**: Delivery rate for GD ads. The curves are smoothed by local polynomial estimator with $degree = 1$, $bandwidth = 0.05$ and Gaussian kernel. The shaded area covers $\pm\sigma$.

the parties:

- The expected revenue of the search engine is the utility of the search engine. Expected revenue is the objective function of the linear program.
- The expected number of clicks is the objective of the advertisers and the users. It is implicitly encoded as constraints of the linear program. It is also a part of the objective function in the regularized version of the algorithm when $\lambda \neq 0$.
- The average cost per click is a different metrics important for the advertisers. We want to make sure that a higher search engine revenue is not a result of higher per-click payments.
- The last performance metric characterizes our ability to satisfy soft click constraints for GD campaigns. Let $\hat{m}_c$ be the number of clicks guaranteed by our solution and m_c be the requested amount. The delivery rate of a single campaign is the ratio of these two quantities not greater than 1 (we do not get benefits for over delivery):

$$rate(c) = \min\left\{1, \frac{\hat{m}_c}{m_c}\right\} \tag{2.42}$$

  Then the micro average delivery rate for all campaigns is:

$$micro\ average\ rate = \frac{1}{\sum_{c \in GD} m_c} \sum_{c \in GD} m_c \cdot rate(c) \tag{2.43}$$

The experimental results are presented on three panels (figure 2.7, figure 2.8 and figure 2.9). Each panel evaluates a different aspect of the problem. Each panel contains four plots with four performance metrics explained above.

On the first panel (figure 2.7) we compare our approach with two baselines. The value of the regularization parameter in this figure is $\lambda = 3$. We repeat the procedure (2.41) 3000 times to get different vectors of auction vs GD decisions. Each decision vector corresponds to one run of the optimization routine. The curves on the plots are mean estimators. We use local polynomial mean estimator with $degree = 1$, $bandwidth = 0.05$ and the Gaussian kernel [69]. The grey shaded area corresponds to $\pm\sigma$ where σ^2 is the variance estimator (obtained by the local polynomial mean estimator for the squared residuals). The average revenue of our approach is comparable to that of the baseline 1, the improvement is 0.2% which is not much, but significant statistically (p-value = 5e-8 for t-test). The number of clicks is greater by 9.1%. This clearly shows the advantage of jointly optimizing GD and auctions. The performance of the baseline 2 is generally worse than that of the baseline 1. This signals us about sub-optimality of straightforward resource distribution between auction and GD mechanisms and again emphasizes the importance of joint optimization. The average CPC of our approach is lower than that of the baseline 1. This indicates that the search engine can further increase its revenue by increasing

the CPC payments for GD campaigns yet keeping the average CPC lower than in the existing system. By changing these premiums the search engine has an effective mechanism of redistributing the benefits of using our framework between itself and the advertisers. The last plot (delivery rate) demonstrates that our approach outperforms the Baseline 2 in its ability to satisfy soft GD constraints. Note, that since the baseline 1 has no guarantee on the delivery, we only make comparison with the baseline 2.

Concluding this comparison, our method is better than both baselines. It is better than the baseline 1 because it has similar revenue, but outperforms the baseline 1 in all the other metrics. Our method is better than the baseline 2 because it significantly outperforms it in three metrics out of four. The only metric in which our method does not win is the average cost per click, mainly, because the baseline 2 has very low revenue. However, the great advantage in the number of clicks and the revenue allows our method to outperform baseline 2 in all four metrics by giving the advertisers a small discount on CPC.

On the second panel (figure 2.8) we investigate the effect of the regularization parameter λ in the objective function:

$$objective = revenue + \lambda \cdot clicks \tag{2.44}$$

Different curves correspond to different values of λ. The effect of λ is monotonic: as the value of the regularizer increases, the number of clicks increases and the revenue decreases. The regularization parameter in the chosen range does not affect the shape of the curves. In the third panel (figure 2.9) we plot the average performance against the value of λ, i.e. each curve in the second panel is averaged and collapsed into one point in the third panel. The values in the third panel (except the delivery rate) are measured in the relative units: the non-regularized performance that corresponds to $\lambda = 0$ is taken to be 1. The number of clicks grows quite fast even for small values of the regularization parameter. The revenue starts to drop at $\lambda = 3$. This observation confirms our assumption, that the search engine revenue is not the best objective of the optimization. We can get much more clicks without losing the revenue if we add advertisers' utility to the objective function.

To sum up, all the above experimental results demonstrate the effectiveness of our proposed joint optimization framework.

2.8 Summary

In this chapter we present the optimization framework that combines pay-per-click auctions and guaranteed delivery (GD) in sponsored search, satisfies advertisers' diverse needs and maximizes the revenue of the search engine. The core of our method is based on linear programming, delayed column generation and dynamic programming. We formulate the optimization problem

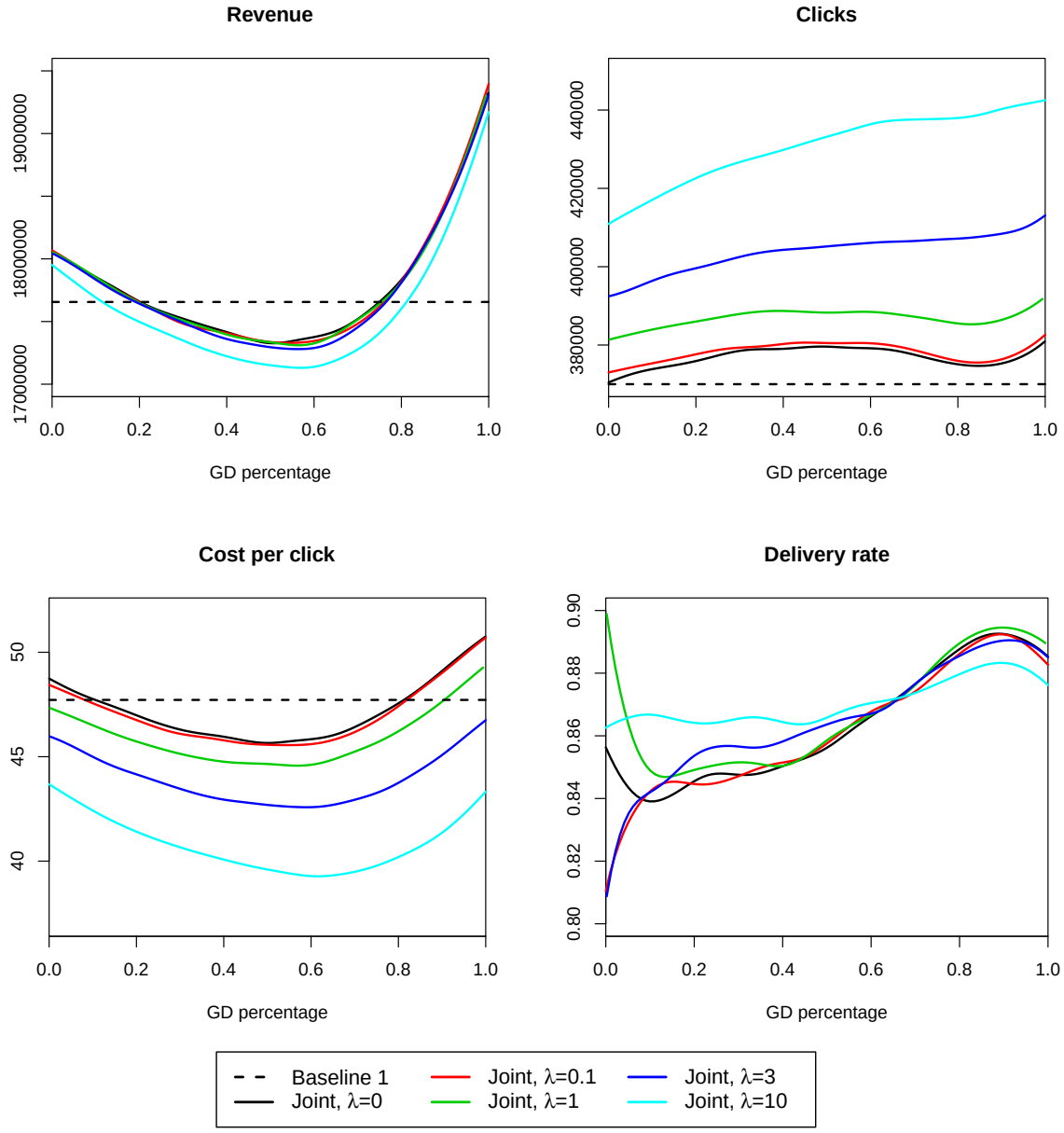


Figure 2.8: **Top Left**: Search engine revenue. **Top Right**: Expected number of clicks. **Bottom Left**: Average cost per click. **Bottom Right**: Delivery rate for GD ads. The curves are smoothed by local polynomial estimator with $degree = 1$, $bandwidth = 0.05$ and Gaussian kernel.

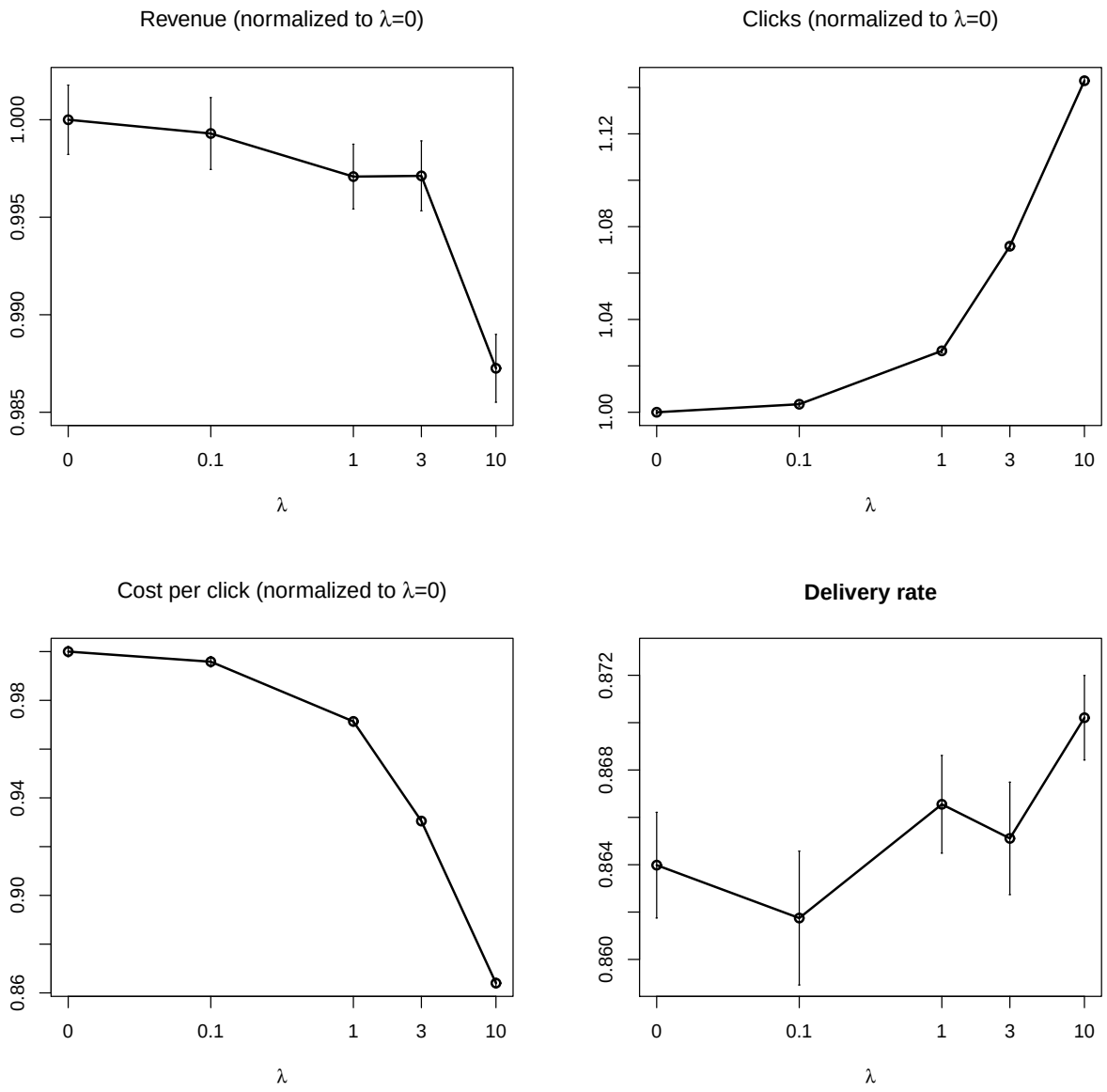


Figure 2.9: **Top Left**: Search engine revenue. **Top Right**: Expected number of clicks (also, proportional to average click trough rate). **Bottom Left**: Average cost per click. **Bottom Right**: Delivery rate for GD ads. Error bars correspond to 95% confidence interval.

with the linear combination of the search engine revenue and the number of clicks as the objective and the advertisers' budgets and the click goals as the constraints. Our experimental findings demonstrate that the proposed approach outperforms the representative baseline methods without joint optimization.

The linear combination of the clicks and the revenue as the objective function aims to better address the needs of the advertisers. This modification provides great benefits compared to the revenue maximization: it allows us to increase the total number of clicks significantly (+7.2%) by sacrificing only 0.3% of the revenue. In the future applications this variant of the framework will provide flexibility in tuning the system performance, especially when other important factors, such as premium payments for GD campaigns, are taken into account.

Finally, in section 2.6 we discuss the adaptation of the algorithm to the online scenario. We cover the following important points: the use of personalized click predictions that become available online; and the problem of unseen data, that is, the small advertisers and tail queries excluded from the optimization.

Chapter 3

Hierarchical Optimization for Sponsored Search

3.1 Introduction

In this chapter we address the computational challenges that arise when the dimensionality of the main optimization problem (2.9) formulated in the previous chapter becomes very large. The dimensionality directly depends on the number of the nodes in the ad allocation graph (Figure 3.1). The nodes on the right hand side of the graph represent campaigns. Their number is not large and does not vary much. The nodes on the left hand side represent the clusters of query submissions. Their number depends on how we group the submissions together and can vary a lot. In the previous chapter each node represents all the submissions of the same query. But we can use a different clustering criterion, for example, one node per submission which leads to billions of nodes. The number of the query nodes characterizes *the granularity* of the allocation problem. The principled approach to model the problem at the right granularity level is the first contribution of the chapter. Specifically, we investigate how the granularity affects the performance and the efficiency of our framework. We show that the granularity levels that are optimal in terms of the performance metrics (revenue and clicks) cannot be handled by the original algorithm efficiently. To address the problem we introduce a new algorithm that is based on hierarchical optimization. This algorithm decomposes the problem into a set of problems of smaller size. These new problems share few common variables and can be solved in parallel independently of each other. The hierarchical optimization algorithm is the second contribution of this chapter.

Why is the granularity so important? Let us take a closer look at the bipartite graph (figure 3.1), that encodes the supply-demand information of our problem. The nodes on the left-hand side of the graph represent query submissions (supply). The nodes on the right-hand side represent advertisements and advertisers' budgets and click goals (demand). The edges represent

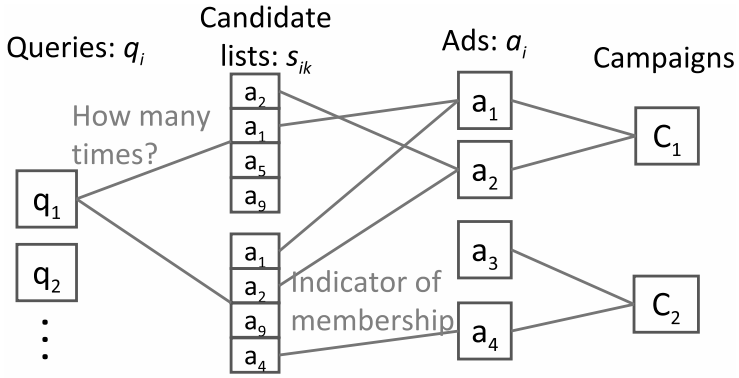


Figure 3.1: Allocation chart

relevance between ads and queries. The edges are assigned the weights: the bids and the click probabilities. In the previous chapter one supply node in the graph corresponds to one unique query string and conveys all the submissions of that query. This choice is convenient because the submissions of the same query have a lot in common: the bids only depend on the keywords and are the same. The click probabilities, on the other hand, are not the same because different submissions of the same query can have very different goals. For example, a user who submits a "digital camera" query can be interested in camera review articles, shooting and developing techniques or might be looking to purchase. The intent behind the query affects the choice of the right ads (just like it affects the optimal organic search results). The search engine can distinguish intent by the search history, the user features and other complementary information [33]. Eventually, it results in different pclick predictions for the ads. If all the submissions of "digital camera" are represented by one node in the graph, we cannot use different click predictions. What can we do? We can model the problem at *finer granularity* by splitting the node into two nodes: all the submissions of "digital camera" with intent to buy and the rest. This leads to a more accurate pclick modeling but also increases the dimensionality of the problem. We can continue splitting until we reach *the finest granularity* level of one node per submission.

What is the optimal granularity in sponsored search? Search engines handle billions of submissions every day therefore the finest granularity (one node per query submission) is too fine. We cannot optimize a graph with billions of nodes. Another reason to avoid very fine granularities is the estimation error for the pclick and the number of submissions. There is no use considering the resolution that is smaller than the estimation error. Finally, very fine granularities reduce the competition between the advertisers which can reduce the revenue of the search engine [43]. On the other hand, the granularity used in the previous chapter is too coarse. Our experimental results demonstrate that finer granularity can significantly improve the revenue and the number of clicks. To model an arbitrary intermediate level of granularity we group the sub-

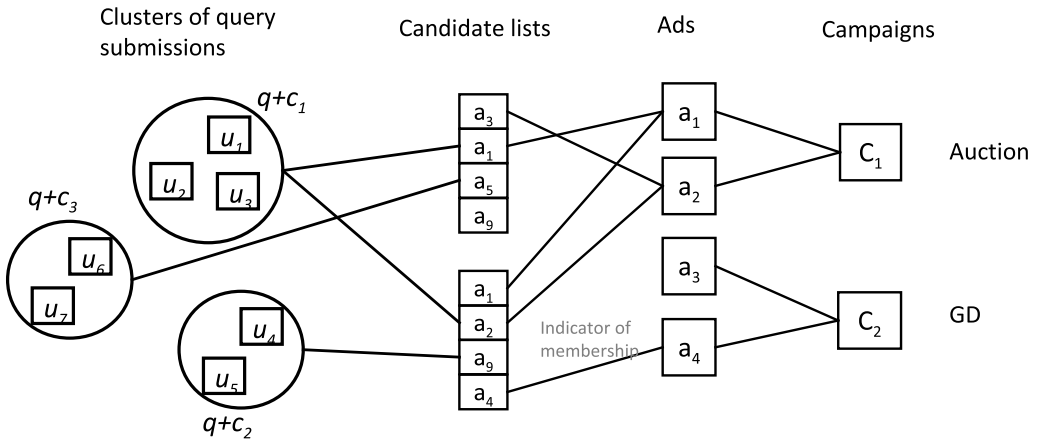


Figure 3.2: The allocation chart with adjusted granularity level. We group the submissions of every query q into clusters c_1, c_2 etc.

missions of a query into coherent clusters using K-means clustering [34] with click predictions as features, figure 3.2. The number of the clusters controls the granularity and the size of the problem. Our goal is to find the optimal granularity level that allows both enough flexibility for the pclick model and efficient optimization. In this work we also propose a method to estimate the upper bound performance that is achievable with perfect pclick predictions and infinite computational resources.

As it turns out, the optimal granularity level that provides good performance is too challenging computationally for the algorithm in chapter 2. The largest problem that we optimize in our experiments has about half a million nodes and a lot more edges. In order to solve the problems of such size we introduce a new optimization algorithm. The idea is to decompose this large optimization problem into multiple small optimization problems that we can solve independently of each other, we call them *children problems*. Essentially, we partition the bi-partite graph from figure 3.2 based on the query string, i.e. all the submissions of the same query with all the relevant ads will go into the same partition. Some ads can be relevant for more than one query and therefore they belong to several partitions simultaneously. To address this problem we introduce query-specific budgets and click requirement for such ads. The query-specific budgets and click requirements form a new set of variables, that we call *the connecting variables*, because they connect the children problems. If we fix the values of the connecting variables we can solve the children problems (allocation of ads for all submissions of one query) independently of each other. To optimize the connecting variables we formulate a special optimization problem, that we call *master optimization*. We prove that the master optimization problem is convex and provide two algorithms to solve it: gradient projection [14, 62] and Dantzig-Wolfe decomposition [20].

In the scientific literature the problem of granularity is mostly studied for display advertising. Display advertising has several distinctive features that make it different from sponsored search. First of all, the advertisers in display advertising can bid on a complex combination of features: age, location, gender, web page etc. The advertisers in sponsored search bid on keywords. The default granularity that perfectly encodes the bidding preferences of the advertisers is determined by the combination of possible feature values. It is very fine grained for display advertising and very coarse for sponsored search. As a result, the researchers in display advertising try to reduce the number of the nodes by merging them together (see channel abstraction model as an example, [68]). Secondly, sponsored search uses the broad match algorithm that can display the ads with the keywords that were not selected by the advertisers. This makes the advertisers' bidding preferences soft. Finally, in display advertising the advertisers are charged per impression and the pclicks are not used. In sponsored search pclick modeling plays the central role in granularity selection. These properties differentiate our work from other works.

In the next section we illustrate the decomposition algorithm with an example, formulate our joint allocation framework as hierarchical optimization and prove its convexity (section 3.2), we provide two algorithms to solve the master optimization problem in sections 3.3 and 3.4. Section 3.5 discusses the clustering of query submissions, the choice of the granularity and the theoretical performance bounds. Finally, we report the experimental results in section 3.6 and conclude the chapter in section 3.7.

3.2 Hierarhical Optimization

3.2.1 Illustrative Example

We start with an illustrative example that fully demonstrates how the hierarchical optimization works. A simple allocation graph on the left of figure 3.3 has three supply nodes and two advertisements. The supply nodes represent two submissions of query q_1 and one submission of query q_2. We assume that both campaigns are GD and the cost per click is 1. The pclicks and the budgets are listed on the figure. One ad is displayed at a time. The optimal allocation is obvious: we display the ad a_1 for the first submission of q_1 and display a_2 for the second submission of q_1 and for the submission of q_2. The expected number of clicks is 3 and the revenue is also 3.

The ad a_2 connects two segments of the graph (queries q_1 and q_2): the budget constraint for a_2 is shared by these segments. Suppose we know that a_2 spends $\$2x$ on q_1 and $\$2(1-x)$ on q_2, i.e. we know the *per-query budget assignment*. Then we can decouple the problem into two independent ones, parameterized by x (right of the figure 3.3). Doing little algebra we get the solution. If $x \in [0, 0.5)$ then there is not enough budget to display a_2 for q_1 and only a_1 will be displayed. If $x = 0.5$ then there is enough budget do display a_2 both for q_1 and q_2. Finally, if $x \in (0.5, 1]$ then there is not enough budget to display a_2 for q_2. The revenue as a function of x

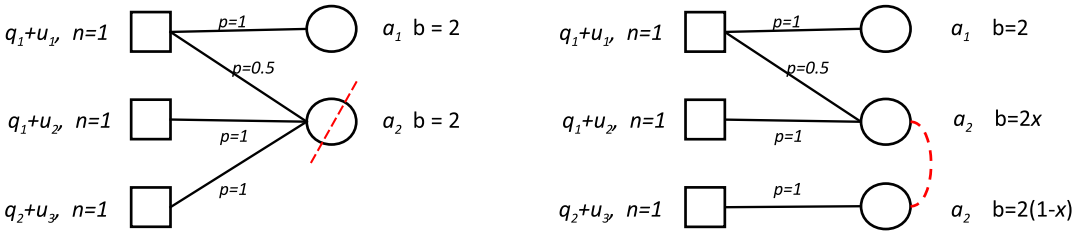


Figure 3.3: Example, domain decomposition. Left: ad a_2 can be displayed for two queries: q_1 and q_2. Right: the problem can be decomposed into two independent problems if we know how much of a_2 budget to assign to query q_1, denoted by $2x$, and how much to assign to query q_2, denoted by $2(1-x)$. Red connection means that the problems are connected by variable x, however, given value of x they are independent.

is:

$$\text{revenue}(x) = \begin{cases} 2 & x \in [0, 0.5) \quad (u_1, u_2, u_3) \to (a_1, -, a_2) \\ 3 & x = 0.5 \quad\;\; (u_1, u_2, u_3) \to (a_1, a_2, a_2) \\ 2 & x \in (0.5, 1] \quad (u_1, u_2, u_3) \to (a_1, a_2, -) \end{cases} \tag{3.1}$$

The last column shows which ad is displayed for each of the query submissions. We can optimize the revenue as a function of x. The optimal revenue is achieved when $x = 0.5$, it leads to the same obvious solution of the problem that we mentioned above.

To make the example more realistic let us scale the problem up (both the numbers of submissions and the budgets) by a large factor N. Large frequencies smooth the solution and make it continuous. Doing simple algebra we get the following solution:

$$\text{revenue}(x) = \begin{cases} 2N(1+x) & x \in [0, 0.5) \\ 3N & x = 0.5 \\ 2N(2-x) & x \in (0.5, 1] \end{cases} \tag{3.2}$$

This revenue function is plotted on figure 3.4. The maximum of this function can be identified graphically: the optimal $x = 0.5$ and the corresponding optimal revenue is $3N$.

The example demonstrates all the essential points of our hierarchical optimization approach:

- We decompose the problem into two independent problems, one per query.
- The children problems are parameterized by x, query specific budget distribution of a_2. We can solve the children problems independently and express the combined revenue/number of clicks as a function of x.
- Optimizing the revenue with respect to x is the master optimization problem. The objective function of the master problem in our example is concave, see figure 3.4. This is not a coincidence: we prove that the master optimization problem is convex.

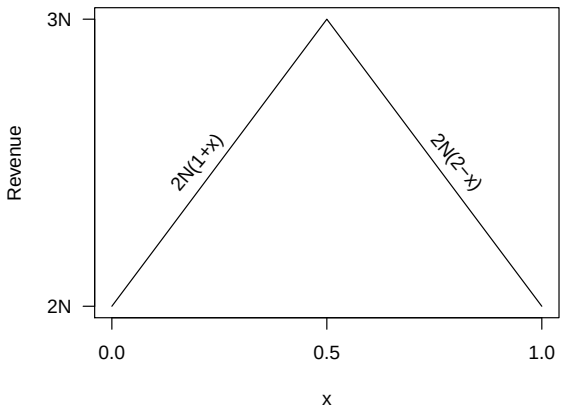


Figure 3.4: Example revenue as a function of x - the proportion of budget of a_2 assigned to the query q_1.

3.2.2 Problem Formulation

In this section we derive the formulation of the hierarchical optimization (specifically, bi-level programming formulation) for auctions and GD in sponsored search. We start with the optimization problem from the previos chapter; extend it to support different granularities; and then we introduce additional variables and decompose the problem into the master and the children optimization problems.

We use the compact vector form (2.14) from the previous chapter:

$$\begin{aligned} \max_{\mathbf{x},\xi_c} \; & \boldsymbol{\alpha}^\intercal \mathbf{x} + \sum_{c \in GD} (d_c - \mu_c \xi_c) \\ \text{s.t.} \quad & \boldsymbol{\delta}_c^\intercal \mathbf{x} \leq d_c \quad \forall c \in A \\ & \boldsymbol{\beta}_c^\intercal \mathbf{x} + \xi_c \geq m_c \quad \forall c \in GD \\ & \mathbf{1}^\intercal \mathbf{x}_i \leq 1 \quad \forall i \\ & \mathbf{x}, \xi_c \geq 0 \end{aligned} \tag{3.3}$$

In this variant of the notation each component of the vector $\mathbf{x}$ has a double index ik, where i denotes the query and k denotes the ranked list of ads.

Introduction of clusters of query submissions into this formulation is quite straightforward. The cluster of submissions in the new formulation plays exactly the same role as a query in the version above, just compare the figures 3.1 and 3.2. Now each left-hand side node, originally

indexed by the subscript i, is indexed by a double script il, where superscript l denotes the query and subscript i denotes cluster within the query. Each cluster of submissions will have its own columns of ads, represented by a variable x^l_{ik}, and its own click probabilities/payments for the relevant ads:

$$\begin{aligned}
&\max_{\mathbf{x}_i,\xi_c} \sum_l \boldsymbol{\alpha}^{l\intercal}\mathbf{x}^l + \sum_{c\in GD}(d_c - \mu_c\xi_c) \\
&\text{s.t.} \quad \sum_l \boldsymbol{\delta}_c^{l\intercal}\mathbf{x}^l \le d_c \quad \forall c \in A \\
&\sum_l \boldsymbol{\beta}_c^{l\intercal}\mathbf{x}^l + \xi_c \ge m_c \quad \forall c \in GD \\
&\mathbf{1}^\intercal\mathbf{x}_i^l \le 1 \quad \forall i,l \\
&\mathbf{x}_i^l, \xi_c \ge 0
\end{aligned} \tag{3.4}$$

We use superscript l to emphasize the summation over queries, it will also be a basis for problem decomposition. The double index ik is still obscured by the vector notation. Essentially, (3.4) differs from (3.3) by one extra summation. We can use the algorithm from the previous chapter directly to solve (3.4). We refer to this variant of the solution as a *flat baseline*.

To decompose the problem we first introduce several redundant variables and rewrite the constraints in the equivalent form:

$$\sum_l \boldsymbol{\delta}_c^{l\intercal}\mathbf{x}^l \le d_c \iff \begin{cases} \sum\limits_l \boldsymbol{\delta}_c^{l\intercal}\mathbf{x}^l \le \sum\limits_l d_c^l \le d_c \\ d_c^l \ge 0 \end{cases} \tag{3.5}$$

d_c^l is a per query budget for auction campaigns. Algebraically, d_c^l is redundant and both variants are equivalent constraints for the variable $\mathbf{x}$. Now we rewrite the constraints in the following way:

$$\begin{cases} \sum\limits_l \boldsymbol{\delta}_c^{l\intercal}\mathbf{x}^l \le \sum\limits_l d_c^l \le d_c \\ d_c^l \ge 0 \end{cases} \iff \begin{cases} \boldsymbol{\delta}_c^{l\intercal}\mathbf{x}^l \le d_c^l \\ \sum\limits_l d_c^l \le d_c \\ d_c^l \ge 0 \end{cases} \tag{3.6}$$

We rewrite GD click constraints using a similar technique with additional variables ξ_c^l and m_c^l:

$$\sum_l \boldsymbol{\beta}_c^{l\intercal}\mathbf{x}^l + \xi_c \ge m_c \iff \begin{cases} \sum\limits_l \boldsymbol{\beta}_c^{l\intercal}\mathbf{x}^l + \sum\limits_l \xi_c^l \ge \sum\limits_l m_c^l \ge m_c \\ m_c^l, \xi_c^l \ge 0 \end{cases} \iff \begin{cases} \boldsymbol{\beta}_c^{l\intercal}\mathbf{x}^l + \xi_c^l \ge m_c^l \\ \sum\limits_l m_c^l \ge m_c \\ m_c^l, \xi_c^l \ge 0 \end{cases} \tag{3.7}$$

Most of the newly introduced variables are zero. If a campaign c is not relevant for a query q_l then the corresponding additional variables d_c^l, m_c^l and ξ_c^l are zero. The problem formulation is now:

$$\max_{\mathbf{x}^l, \xi_c^l, m_c^l, d_c^l} \sum_l \left[\boldsymbol{\alpha}^{l\intercal} \mathbf{x}^l - \sum_{c \in GD} \mu_c \xi_c^l \right] + \sum_{c \in GD} d_c \tag{3.8}$$

$$\text{s.t.} \quad \boldsymbol{\delta}_c^{l\intercal} \mathbf{x}^l \leq d_c^l \quad \forall c \in A, \forall l \tag{3.9}$$

$$\boldsymbol{\beta}_c^{l\intercal} \mathbf{x}^l + \xi_c^l \geq m_c^l \quad \forall c \in GD, \forall l \tag{3.10}$$

$$\mathbf{1}^\intercal \mathbf{x}_i^l \leq 1 \quad \forall i, l \tag{3.11}$$

$$\sum_l d_c^l \leq d_c \quad \forall c \in A \tag{3.12}$$

$$\sum_l m_c^l \geq m_c \quad \forall c \in GD \tag{3.13}$$

$$\mathbf{x}^l, \xi_c^l, m_C^l, d_C^l \geq 0$$

We do not need an explicit constraint $\sum_l \xi_c^l \leq \xi_c$. This condition will be satisfied automatically because slack variables are always minimized to be as tight as possible. The solutions of (3.8) and (3.4) are the same. Let us examine the structure of (3.8). The objective is decomposed into blocks, indexed by l. The first three sets of constraints (3.9),(3.10) and (3.11) also have block structure: only the variables with the same superscript l appear in one constraint. The rest constraints (3.12) and (3.13) only contain d_c^l and m_c^l variables. If we fix the values of d_c^l and m_c^l then the problem fully decomposes into l independent optimization problems for x_{ik}^l and ξ_c^l variables, each of these child problems has the form of our main formulation of the previous chapter, i.e. (3.3), with all parameters and variables having the same subscript l. We can rewrite (3.3) problems using the following compact notation:

$$\max_{\mathbf{x}^l, \xi_c^l} f^l(\mathbf{x}^l, \xi_c^l, m_c^l, d_c^l) \tag{3.14}$$

$$\text{s.t.} \quad \{\mathbf{x}^l, \xi_c^l\} \in S^l(m_c^l, d_c^l)$$

where f^l denotes the objective function, parameterized by m_c^l and d_c^l and S^l is a convex feasibility region defined by the original constraints also parameterized by m_c^l and d_c^l. The solution of (3.14) is a function of parameters m_c^l and d_c^l:

$$f^{l*}(m_c^l, d_c^l) = \max_{\mathbf{x}^l, \xi_c^l} f^l(\mathbf{x}^l, \xi_c^l, m_C^l, d_c^l | \{\mathbf{x}^l, \xi_c^l\} \in S^l) \tag{3.15}$$

Finally we can rewrite (3.8) using our new compact notation to fully reveal its hierarchical struc-

ture:

$$\max_{m_c^l, d_c^l} \sum_l f^{l*}(m_c^l, d_c^l) + \sum_{c \in GD} d_c \tag{3.16}$$
$$\text{s.t.} \quad \sum_l d_c^l \leq d_c \quad \forall c \in A$$
$$\sum_l m_c^l \geq m_c \quad \forall c \in GD$$
$$m_c^l, d_c^l \geq 0$$

This is one of the standard hierarchical optimization formulations, also called bi-level mathematical programming problem [4]. The elements of the objective function are optimization problems. The variables m_c^l and d_c^l are called connecting variables. Optimization of connecting variables (as defined in (3.16)) is called the master problem. The optimization problems in $\mathbf{x}$ and ξ variables (as defined in (3.14)) are called child problems.

This decomposition converts a high-dimensional flat mathematical programming problem (3.4) into a lower dimensional hierarchical problem with potentially more complex functional dependency. The convexity theorem from [12] guarantees that the master problem is convex (discussed in the next section). This property allows us to use mixed strategies for optimization with respect to connecting variables and child-problem variables (sections 3.3 and 3.4).

3.2.3 Convexity of the Master Problem

Before proving the convexity of the master problem (3.16) we formulate several complementary lemmas.

Lemma 3.2.1 *Linear program in standard form:*

$$\begin{aligned} &\max \mathbf{c}^\intercal \mathbf{x} \\ \textit{s.t.} \quad &A\mathbf{x} \leq \mathbf{b} \\ &\mathbf{x} \geq 0 \end{aligned} \tag{3.17}$$

can be equivalently rewritten as

$$\max \mathbf{c}^\intercal \mathbf{x} + \sum_i g_+(b_i - \mathbf{a_i}^\intercal \mathbf{x}) \tag{3.18}$$
$$\textit{s.t.} \quad \mathbf{x} \geq 0$$

where $g_+(x)$ is a barrier function:

$$g_+(x) = \begin{cases} 0 & \textit{if} \quad x \geq 0 \\ -\infty & \textit{if} \quad x < 0 \end{cases} \tag{3.19}$$

Proof If any of the constraints is violated then the barrier and therefore the objective functions become $-\infty$. If all the constraints are satisfied, then the objective in (3.18) becomes the original objective $\mathbf{c}^\intercal\mathbf{x}$. ∎

Lemma 3.2.2 *The barrier function $g_+(x)$ is concave.*

Proof Follows directly from the definition of a concave function

$$g_+(\lambda x + (1-\lambda y)) \geq \lambda g_+(x) + (1-\lambda)g_+(y) \quad \lambda \in (0,1) \tag{3.20}$$

if both $x \geq 0$ and $y \geq 0$ then both sides of the inequality are 0 and it holds; if either $x < 0$ or $y < 0$ then the right hand side is $-\infty$ and the inequality holds. ∎

The next lemma is a well-known fact about operations that preserve convexity and goes without proof (see [11]). We will also use other criteria such as composition with affine function and nonnegative weighted sum.

Lemma 3.2.3 (Maximization criteria) *if $f(x,y)$ is concave in (x,y) and C is a convex set, then*

$$g(x) = \inf_{y \in C} f(x,y) \tag{3.21}$$

is concave.

Lemma 3.2.4 *The solution of a standard LP* (3.17) *is concave as a function of its parameter* $\mathbf{b}$.

Proof According to the Lemma 3.2.1 we can reformulate the standard LP as (3.18). The objective of reformulation:

$$f(\mathbf{x}, \mathbf{b}) = \mathbf{c}^\intercal\mathbf{x} + \sum_i g_+(b_i - \mathbf{a_i}^\intercal\mathbf{x}) \tag{3.22}$$

is concave in $(\mathbf{x}, \mathbf{b})$ as a sum of a linear function $\mathbf{c}^\intercal\mathbf{x}$ and a composition of concave g_+ (Lemma 3.2.2) and affine $b_i - \mathbf{a_i}^\intercal\mathbf{x}$ functions. Now we can apply maximization criteria lemma 3.2.3 to $f(\mathbf{x}, \mathbf{b})$. ∎

Theorem 3.2.5 (Convexity of master problem) *The problem defined in* (3.16) *is convex.*

Proof The constrained maximization problem is concave if and only if the feasibility region is convex and the objective function is concave (convex for minimization problems). The convexity of the feasibility region is obvious as it is defined by the set of linear inequalities. The objective function is a summation, it is sufficient to establish concavity of each summand:

$$f^{l*}(m_C^l, d_C^l) = \max_{x_{ik}^l, \xi_C^l} f^l(x_{ik}^l, \xi_C^l, m_C^l, d_C^l | \{x_{ik}^l, \xi_C^l\} \in S^l) \tag{3.23}$$

here f^* is a solution of an LP and its concavity follows directly from the Lemma 3.2.4. ∎

3.3 Solving the Master Problem via Gradient Projection

The problem is (3.16) is a standard convex optimization problem with linear constraints.

$$\begin{aligned} \max_{m_c^l, d_c^l} & \sum_l f^{l*}(m_c^l, d_c^l) + \sum_{c \in GD} d_c \\ \text{s.t.} \quad & \sum_l d_c^l \leq d_c \quad \forall c \in A \\ & \sum_l m_c^l \geq m_c \quad \forall c \in GD \\ & m_c^l, d_c^l \geq 0 \end{aligned} \tag{3.24}$$

The components of the objective function (solutions of child problems) are monotone with the arguments: as d_c^l increases for auction campaigns or m_c^l decreases for GD campaigns the objective function is guaranteed not to decrease. Essentially this means that providing more money for Auction campaigns and decreasing click burden for GD campaigns cannot make the allocation worse. It follows that there exists a solution of (3.24) with all the constraints active and inequalities can be replaced with equalities.

The most generic optimization method available to solve (3.24) is a gradient projection method [14]. To be precise, we use a subgradient projection [62] because the solution of a linear program is a convex polytope of the parameters and is not differentiable. The gradient projection method, as its name suggests, projects the gradient of the objective function onto the feasibility region. The projection yields the direction of the steepest ascent within the feasibility region. The algorithm is presented below:

1: Solve $\max_{\mathbf{x}} f(\mathbf{x})$ subject to $A\mathbf{x} \leq \mathbf{b}$
2: Find initial feasible solution $\mathbf{x}_0$: $A\mathbf{x}_0 \leq \mathbf{b}$, set step $n = 0$
3: Initialize the subset of active constraints J: $\mathbf{a}_j^\intercal \mathbf{x} = b_j$ for $j \in J$
4: **while** $|f(\mathbf{x}_{n-1}) - f(\mathbf{x}_n)| > \epsilon$ - not converged **do**
5: Compute gradient: $\nabla f(\mathbf{x}_n)$
6: The search direction is the orthogonal projection of the gradient onto the subspace of active constraints: $\mathbf{s} = [I - A_J^\intercal (A_J A_J^\intercal)^{-1} A_J] \nabla f$
7: if $\mathbf{s} = 0$ we need to check the vector of Lagrange multipliers: $\lambda = (A_J A_J^\intercal)^{-1} A_J \nabla f$. If $\lambda_j \leq 0 \ \ \forall j \in J$ we found maximum, if some $\lambda_j > 0$ we should remove a constraint j from the active set J and recompute the search direction.
8: Do linear search: $\alpha^* = \arg\max_{\alpha > 0} f(\mathbf{x}_n + \alpha \mathbf{s})$. The possible value of α has an upper bound, that is defined by non-active constraints - i.e. we can move forward until some of the inactive constraints becomes active, if this point is the maximum we add the constraint to the set of active constraints J.

9: Update: $\mathbf{x}_{n+1} := \mathbf{x}_n + \alpha^* \mathbf{s}$ and $n := n + 1$.
10: **end while**

We use this algorithm to solve the problem (3.24). We derive all the equations of the algorithm (updates, projections and Lagrange multipliers) for our problem: most steps take very simple and intuitive forms. Below we explain how some of the steps of the algorithm look for our problem.

Step 2. We use a non-hierarchical solution with coarse granularity, i.e. the solution from the first chapter, to initialize a feasible starting point $\mathbf{x}_0$. We compute how much money each advertisers spends per query according to the coarse solution and use it as d_c^l. If the advertiser does not spend all of his budget, $\sum\limits_{c \in GD} d_c^l < d_c$, we scale query-specific budgets d_c^l up until the budget constraint is active. We do the same for the clicks of GD campaigns m_c^l. In case of underdelivery $\sum\limits_{c \in GD} m_c^l < m_c$ we scale all m_c^l up until $\sum\limits_{c \in GD} m_c^l = m_c$.

Step 3. The budget and the click constraints in the solution are active. Some of the positivity constraints can be active too. Since our variables are grouped by campaigns, we group active positivity constraints by campaigns as well. J_c: $l \in J_c \iff d_c^l = 0$ for auctions or $m_c^l = 0$ for GD.

Step 5. To compute the gradient (subgradient) of the objective function we need to compute the gradient (subgradient) of each component f^{l^*} which is a solution of an LP with respect to the parameters of the constraint matrix. We can use the vector of dual variables of the child as its gradient direction. The dual variable y_j for a constraint $\mathbf{a}_j^\intercal \mathbf{x} \leq b_j$ tells us the improvement of the objective if we are allowed to relax b_j. If the dual variable $y_j = 0$ then the constraint is not active and relaxing it won't change the objective. Let us denote the gradients of the original problem as:

$$y_c^l = \begin{cases} \frac{\partial}{\partial d_c^l} f_l^* & c \in A \\ \frac{\partial}{\partial m_c^l} f_l^* & c \in GD \end{cases} \tag{3.25}$$

In the previous chapter we used y_c to denote the dual variable of the LP, that corresponds to the campaign budget or click constraint, here it denotes the same dual variable but for the child problem l.

Step 6. The search direction computation. Our constraints have block diagonal form, each variable has only one non-zero coefficient in the constraints matrix (plus positivity constraint), this makes the computation of the projection very easy. For every children task l and for every campaign c:

$$s_c^l = \begin{cases} y_c^l - mean_{j \notin J_c}(y_c^j) & \text{if} \quad l \notin J_c \\ 0 & \text{if} \quad l \in J_c \end{cases} \tag{3.26}$$

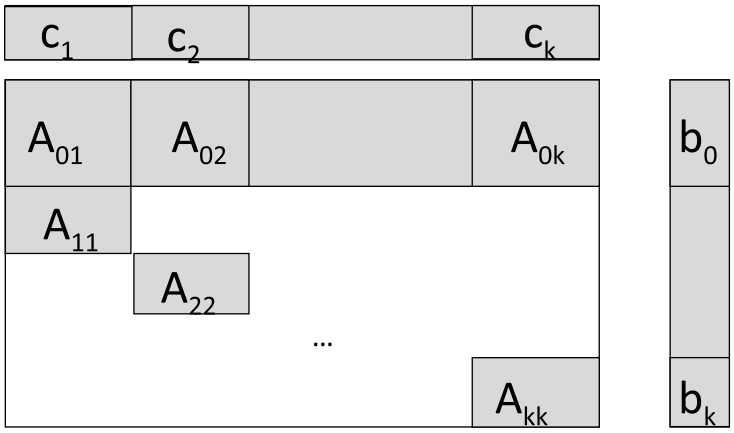


Figure 3.5: Block angular structure of LP. Grey blocks correspond to non-zero variables; white blocks correspond to zero variables. The vector $\mathbf{c}_i$ is a subvector of the objective coefficients vector, A is the constraints matrix, $\mathbf{b}_i$ is the subvector of the constraints bound.

The intuition about the above expression is the following. We increase the variables with partial derivative above average and decrease the variables with partial derivative below average. This corresponds to the redistribution of a limited resource. Since the sum of resources of each campaign is fixed, we can only add resources to a child by taking them from other children. The resources go to those children that use the campaign resources more efficiently (higher y_c^l). The second line tells us that those variables (active constraints) that are set to zero remain constant.

Step 7. If $\mathbf{s} = 0$ we should check if we can improve the objective by relaxing some of the active constraints. The candidates are:

$$l \in J_c : y_c^l > mean_{j \notin J_c}(y_c^j) \tag{3.27}$$

Following the intuition of the step 6, if some variable has a good potential to increase the objective function we should remove it from the set of active constraints.

Step 8. In this step we identify the upper limit α_{max} of the linear search. This limit is determined by the first constraint that becomes active. In our formulation this is the first non-zero variable that becomes zero. For convenience we rename the variables as $z_c^l = d_c^l$ if $c \in A$ and $z_c^l = m_c^l$ if $c \in GD$, then:

$$z_c^l + \alpha s_c^l = 0 \quad \alpha > 0 \tag{3.28}$$

$$\alpha_{max} = \min_{c,l:s_c^l<0} -z_c^l/s_c^l \tag{3.29}$$

3.4 Solving the Master Problem via Dantzig-Wolfe Decomposition

In this section we present a decomposition technique that is designed especially for linear programs. The original flat optimization problem (3.8) in chapter 2 is a linear program. Hierarchical decomposition of (3.8) results in a set of children subprograms (also LP) and the master program (not LP). Is it possible to decompose the problem in such a way that the master is also a linear program? The answer is yes: the programs with special structure can be solved using Dantzig-Wolfe decomposition [20].

Let us summarize the main idea of the algorithm first: the solution of the master program can be represented as a linear combination of the solutions of the subproblems. The master program requests an additional solution of a subproblem if it can improve the master's objective. To request a new solution of a subproblem the master selects the parameter values for this subproblem. This algorithm is based on delayed column generation. In chapter 2 we already used delayed column generation to solve the linear program formulated for the joint optimization framework. The foundation for the application of DCG here is the same: few constraints and a huge number of varialbes. To select (generate) the important variables we formulate a complementary optimization problem. In chapter 2 these complementary problems are solved by dynamic programming. In this section the complementary problems are formulated linear programs.

Now we can proceed to the formal description of the algorithm. In order to apply Dantzig-Wolfe decomposition the program has to have the block angular structure, see figure 3.5:

$$\begin{aligned} \max_x \mathbf{c}_1^\intercal \mathbf{x}_1 + \cdots + \mathbf{c}_k^\intercal \mathbf{x}_k & \qquad (3.30) \\ A_{01}\mathbf{x}_1 + \cdots + A_{0k}\mathbf{x}_k &\le \mathbf{b}_0 \\ A_{11}\mathbf{x}_1 &\le \mathbf{b}_1 \\ \cdots & \\ A_{kk}\mathbf{x}_k &\le \mathbf{b}_k \\ \mathbf{x}_i &\ge 0 \end{aligned}$$

The first set of constraints, expressed by A_{0i} may contain all the variables of the problem. These constraints are called linking constraints. Other constraints segment the variables into k subsets and form a block-diagonal constraints matrix. Without linking constraints A_{0i} the optimization problem can be fully decomposed into k independent optimization problems with each subproblem covering its subset of the variables.

We will first outline the Dantzig-Wolfe decomposition algorithm that solves (3.30) and then derive the equations for our problem (3.8). The algorithm is based on the representation theorem for convex polyhedra. Convex polyhedron is the intersection of a finite set of linear equalities

and/or inequalities. Feasibility region of a linear program is a convex polyhedron. The representation theorem states that every bounded convex polyhedron can be expressed as a convex combination of its extreme points (vertices), or in other words, a convex hull of its extreme points. For unbounded polyhedra the theorem is extended in the following way: the convex polyhedron can be expressed as a convex combination of its extreme points and a non-negative combination of its extreme rays. Below are the two alternative representations of a polyhedron X_i according to the representation theorem:

$$X_i: \quad \begin{cases} A_{ii}\mathbf{x}_i \leq \mathbf{b}_i \\ \mathbf{x}_i \geq 0 \end{cases} \iff \begin{cases} \mathbf{x}_i = \sum_j \lambda_{ij}\mathbf{v}_{ij} + \sum_j \mu_{ij}\mathbf{r}_{ij} \\ \sum_j \lambda_{ij} = 1 \\ \lambda_{ij} \geq 0, \quad \mu_{ij} \geq 0 \end{cases} \tag{3.31}$$

where $\mathbf{v}_{ij}$ is a vertex (extreme point) of the convex polyhedron X_i, $\mathbf{r}_{ij}$ is its extreme ray (extreme direction) and λ_{ij} and μ_{ij} are the coefficients of the linear combination. We apply this theorem to the constraints of (3.30) and rewrite it in a compact form as:

$$\begin{aligned} \max_x \mathbf{c}_1^\intercal \mathbf{x}_1 + \cdots + \mathbf{c}_k^\intercal \mathbf{x}_k \\ A_{01}\mathbf{x}_1 + \cdots + A_{0k}\mathbf{x}_k \leq \mathbf{b}_0 \\ \mathbf{x}_i \in X_i \end{aligned} \tag{3.32}$$

or, using the right hand side of (3.31) as:

$$\begin{aligned} \max_{\lambda,\mu} \sum_{i=1}^{k} \mathbf{c}_i^\intercal \left[\sum_j \lambda_{ij}\mathbf{v}_{ij} + \sum_j \mu_{ij}\mathbf{r}_{ij} \right] \\ \sum_{i=1}^{k} A_{0i} \left[\sum_j \lambda_{ij}\mathbf{v}_{ij} + \sum_j \mu_{ij}\mathbf{r}_{ij} \right] \leq \mathbf{b}_0 \\ \sum_j \lambda_{ij} = 1 \quad \forall j \in 1..k \\ \lambda_{ij} \geq 0, \quad \mu_{ij} \geq 0 \end{aligned} \tag{3.33}$$

This reformulation is also a linear program with new variables λ and μ, each new variable corresponds to a vertex or to an extreme ray of one of the k polyhedra. The number of vertices/rays can be very large, but the number of constraints is small: $dim(\mathbf{b}_0) + k$. This means that the number of non-zero variables in the solution is limited. We use delayed column generation to identify these good variables (vertices and rays). (3.33) above is an unrestricted master problem that is equivalent to the original LP (3.30). At each step we assume that only subset of vertices

and rays for each of the k polytops is present in the formulation, we denote these subsets as $J_i^{(v)}$ (for vertices) and $J_i^{(r)}$ for ray. The problem below differs from (3.33) only in the summation limits for j:

$$\max_{\lambda,\mu} \sum_{i=1}^{k} \mathbf{c}_i^\intercal \left[\sum_{j \in J_i^{(v)}} \lambda_{ij} \mathbf{v}_{ij} + \sum_{j \in J_i^{(r)}} \mu_{ij} \mathbf{r}_{ij} \right] \tag{3.34}$$

$$\sum_{i=1}^{k} A_{0i} \left[\sum_{j \in J_i^{(v)}} \lambda_{ij} \mathbf{v}_{ij} + \sum_{j \in J_i^{(r)}} \mu_{ij} \mathbf{r}_{ij} \right] \leq \mathbf{b}_0 \tag{3.35}$$

$$\sum_{j} \lambda_{ij} = 1 \quad \forall i \in 1..k \tag{3.36}$$

$$\lambda_{ij} \geq 0, \quad \mu_{ij} \geq 0$$

This is the restricted version of the master problem (since we removed some of the variables). The number of variables that we keep is equal to the maximum possible rank of the constraints matrix: $\sum_{i=1}^{k} |J_i^{(v)}| + |J_i^{(r)}| = dim(\mathbf{b}_0) + k$. Now we will iteratively add some of the removed variables that can improve the objective. A missing vertex $\mathbf{v}_{ij}$ will improve the objective of (3.34) if and only if the corresponding variable λ_{ij} has positive reduced price:

$$\mathbf{c}_i^\intercal \mathbf{v}_{ij} - \mathbf{y}^\intercal A_{0i} \mathbf{v}_{ij} - z_i > 0 \tag{3.37}$$

where $\mathbf{y}$ is a vector of dual variables that correspond to the constraints (3.35) and z_i is a dual variable that corresponds to the i-th constraint of (3.36). To find the variable with the positive reserved price we can formulate the following maximization problem:

$$\max_{\mathbf{v}_{ij}} (\mathbf{c}_i^\intercal - \mathbf{y}^\intercal A_{0i}) \mathbf{v}_{ij} \tag{3.38}$$

$$\text{s.t} \quad \mathbf{v}_{ij} \text{ is a vertex of polyhedron } X_i$$

Now we reverse the representation theorem to go back from convex hull representation to the inequalities representation:

$$\max_{\mathbf{x}_i} (\mathbf{c}_i^\intercal - \mathbf{y}^\intercal A_{0i}) \mathbf{x}_i \tag{3.39}$$

$$\text{s.t} \quad A_{ii} \mathbf{x}_i \leq \mathbf{b}_i$$

$$\mathbf{x}_i \geq 0$$

This optimization problem is our child optimization. If we go back and take a look at the structure of the original problem (3.30) we can see that the child corresponds to a block diagonal

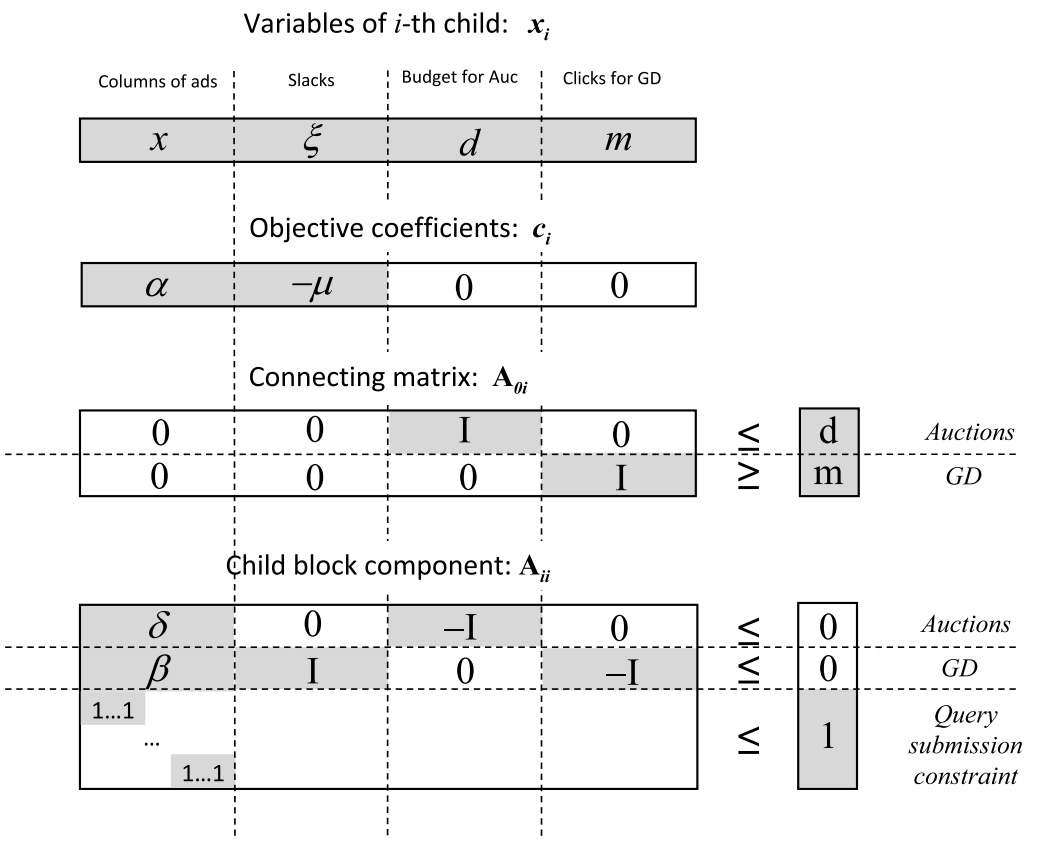


Figure 3.6: Mapping of our LP formulation to generic Dantzig-Wolfe decomposition notation. Variables $\mathbf{x}_i$, objective coefficients $\mathbf{c}_i$, connecting matrix A_{0i}, child specific matrix A_{ii} and the vector of constraint bounds $\mathbf{b}_0$, $\mathbf{b}_i$ are presented.

component i with the modified objective. If the solution of (3.39) is bounded and the objective is $> z_i$ then we add this solution as a new vertex to the restricted master problem (3.34). If the objective is $\leq z_i$ we terminate the optimization algorithm. If (3.39) is unbounded, then we have a candidate ray to add the master problem.

Let us apply this technique to our linear program (3.8). Specifically, we will formulate the master and the child problems. First of all we want to make sure that our problem is block angular and specify every element of the decomposition: A_{0i}, A_{ii}, $\mathbf{b}_0$, $\mathbf{b}_i$, $\mathbf{c}_i$, $\mathbf{x}_i$. This is easy to do graphically. On Figure 3.6 we show all the variables and all the constraints of (3.8) groupped in block angular form.

The variables for each child consist of four sets: the variables that correspond to the columns of ads (x), the slack variables for GD campaigns (ξ), the child specific auction budget and GD

click constraint variables (d and m). The objective function contains only ad column variables and the slack variables. The connecting constraints contain only child specific budgets and click requirements:

$$\sum_l d_c^l \leq d_c \tag{3.40}$$

$$\sum_l m_c^l \geq m_c \tag{3.41}$$

$$\tag{3.42}$$

I.e. the sum of child specific budgets of a campaign cannot exceed the global campaign budget. Similarly, the click requirements of all the children cannot be lower than the global click requirement. The blocks A_{ii} represent within child constraints for spending, clicks and number of displays. We can formulate our variant of the child problem l (3.39):

$$\begin{aligned}
\max_{\mathbf{x}^l, \xi^l, \mathbf{m}^l, \mathbf{d}^l} \quad & \boldsymbol{\alpha}^{l\mathsf{T}} \mathbf{x}^l - \sum_{c \in GD} \mu_c \xi_c^l + \sum_{c \in GD} y_c m_c^l - \sum_{c \in A} y_c d_c^l \\
\text{s.t.} \quad & \boldsymbol{\delta}_c^{l\mathsf{T}} \mathbf{x}^l \leq d_c^l \quad \forall c \in A \\
& \boldsymbol{\beta}_c^{l\mathsf{T}} \mathbf{x}^l + \xi_c^l \geq m_c^l \quad \forall c \in GD \\
& \sum_k x_{ik}^l \leq 1 \quad \forall i \\
& x_{ik}^l, \xi_c^l, m_c^l, d_C^l \geq 0
\end{aligned} \tag{3.43}$$

where $y_c \geq 0$ are dual variables of the master problem. Pay attention, that the budgets d_c and the clicks requirements m_c and variables. This means that we can set the budget as large as we want. Of course, using a large budget is penalized in the objective function: money is a resource that has its cost. We cannot use very large budgets efficiently because the number of query submissions is limited. Finding the most efficient budgets becomes a part of the optimization problem. The same reasoning applies to the click requirements of GD campaigns.

Fortunately, we can find the optimal ξ_c^l, m_c^l and d_c^l analytically. First, dual variables $y_c \geq 0$, therefore smaller values of d_c^l and larger values of m_c^l favour the objective and the constraints are active:

$$\begin{aligned}
\boldsymbol{\delta}_c^{l\mathsf{T}} \mathbf{x}^l &= d_c^l \quad \forall c \in A \\
\boldsymbol{\beta}_c^{l\mathsf{T}} \mathbf{x}^l + \xi_c^l &= m_c^l \quad \forall c \in GD
\end{aligned} \tag{3.44}$$

We can substitute these equalities into the objective:

$$\left(\boldsymbol{\alpha}^{l\mathsf{T}} + \sum_{c \in GD} y_c \boldsymbol{\beta}_c^{l\mathsf{T}} - \sum_{c \in A} y_c \boldsymbol{\delta}_c^{l\mathsf{T}} \right) \mathbf{x}^l + \sum_{c \in GD} (y_c - \mu_c) \xi_c^l \tag{3.45}$$

If $y_c > \mu_c$ the problem is unbounded, the solution is $\xi_c^l = +\infty$. This solution of a child problem corresponds to an extreme ray in the direction of positive ξ_c^l axis. The number of rays is constrained by the number of GD campaigns. If $y_c \leq \mu_c$ then $\xi_c^l = 0$ is the solution and the problem reduces to:

$$\max_{\mathbf{x}^l \geq 0} \left(\boldsymbol{\alpha}^{l\mathsf{T}} + \sum_{c \in GD} y_c \boldsymbol{\beta}_c^{l\mathsf{T}} - \sum_{c \in A} y_c \boldsymbol{\delta}_c^{l\mathsf{T}} \right) \mathbf{x}^l \qquad (3.46)$$
$$\text{s.t.} \quad \sum_k x_{ik}^l \leq 1 \quad \forall i$$

The solution of this problem is quite obvious. For each cluster of submissions i we pick one variable with the highest objective coefficient and set it to one. The problem of finding the highest coefficient is equivalent to the problem of finding the best column in chapter 2. We solve it using the same dynamic programming algorithm:

$$\max \left(\alpha + \sum_{c \in GD} y_c \beta_c - \sum_{c \in A} y_c \delta_c \right) \qquad (3.47)$$

To summarize the algorithm presented in this section. The flat optimization has individual ranked lists of ads as variables. The decomposition technique groups several lists together to form a high level object. Each object has one ranked list per each cluster of submissions for some query. The goal of the child optimization is to generate these objects. The goal of the master optimization is to combine these objects together.

3.5 Clustering of Query Submissions

To control the granularity of the allocation problem we group the query submissions into coherent clusters. The granularity is determined by the size and the number of the clusters - we are free to choose these parameters. Each cluster represents a node in the allocation chart and therefore all members of the cluster share the same allocation parameters, most importantly, click probability predictions. This gives the intuition which clustering criteria to use: the query submissions with similar ad pclicks should be clustered together. We use pclicks as features of the query. For every query q the search engine determines a set of relevant ads J_q. For every submission q_i of query q the search engine estimates the click probability of these relevant ads:

$$p_{ij} = \begin{cases} PClick(q_i, a_j) & j \in J_q \\ 0 & j \notin J_q \end{cases} \qquad (3.48)$$

Each query submission q_i is represented by the vector of pclick predictions $\mathbf{p}_i$. We use K-means [34] to cluster the submissions. The minimization objective of K-means is the within cluster sum of squares (WCSS):

$$\arg\min_{\mathbf{S}} \sum_{l=1}^{K} \sum_{q_i \in S_l} ||\mathbf{p_i} - \overline{\mathbf{p}}_l||^2 \tag{3.49}$$

where $\overline{\mathbf{p}}_l$ is the mean of the cluster l

$$\overline{\mathbf{p}}_l = \frac{1}{|S_l|} \sum_{q_i \in S_l} \mathbf{p_i} \tag{3.50}$$

K-means tends to produce the clusters of equal size. The quality of the allocation should increase monotonically with the number of clusters K. In the limit case, when K is equal to the number of distinct submissions, we get the perfect allocation (under the condition that the pclick predictions are accurate). Different criteria can be used to select the number of clusters. The problem size and execution time grows with K therefore we can select K based on acceptable execution time. A principled way to select K is based on the size of the cluster. Since pclick predictions and therefore feature vectors $\mathbf{p}_i$ are not perfect, making cluster size too small, i.e. comparable with the estimation error, does not make sense. This criterion allows us to choose the cluster size adaptively for each query.

In this work we do not investigate the effect of feature selection and feature scaling. These techniques can potentially improve the clustering quality. For example, the advertisements with small budgets should have reduced effect on the clustering since they are not displayed very often. On the other hand the weight of the ads with high bids (for auctions) or high per click costs (for GD) should be increased. We leave this analysis for the future work.

Below we provide some intuition about the convergence of the objective function of the allocation (search engine revenue) as the number of clusters increases. For simplicity we assume that there is only one query q. We also assume that we can express the probability of the ad a_j to be displayed times the average payment of this ad as a function of click probabilities of all the competitors: $f_j(\mathbf{p})$. Then the expected payment for this ad if the query q is submitted is $f_j(\mathbf{p})p_j$. Similarly, the expected revenue from all the ads given one submission of the query is:

$$g(\mathbf{p}) = f^{\intercal}(\mathbf{p})\mathbf{p} = \sum_j f_j(\mathbf{p})p_j \tag{3.51}$$

For a number of submissions of this query the expected revenue is:

$$R = \sum_i f^{\intercal}(\mathbf{p_i})\mathbf{p_i} \tag{3.52}$$

Now assume that we group all the submissions of the query into several clusters with means $\overline{\mathbf{p}}_l$. The probability of displaying the ad and the payment depends only on the cluster means, and then the modified revenue is:

$$R_K = \sum_{l=1}^{K} \sum_{i \in S_l} f^{\intercal}(\overline{\mathbf{p}}_l)\mathbf{p_i} = \sum_{l=1}^{K} f^{\intercal}(\overline{\mathbf{p}}_l) \sum_{i \in S_l} \mathbf{p_i} = \sum_{l=1}^{K} |S_l| f^{\intercal}(\overline{\mathbf{p}}_l)\overline{\mathbf{p}}_l \tag{3.53}$$

The difference of revenues:

$$R - R_K = \sum_{l=1}^{K} \sum_{i \in S_l} (f^{\intercal}(\mathbf{p}_i)\mathbf{p}_i - f^{\intercal}(\overline{\mathbf{p}}_l)\overline{\mathbf{p}}_l) = \sum_{l=1}^{K} \sum_{i \in S_l} (g(\mathbf{p}_i) - g(\overline{\mathbf{p}}_l)) \tag{3.54}$$

Let's assume that $g(\mathbf{p})$ is Lipschitz continuous, i.e.

$$\exists C \in R: \quad |g(\mathbf{p}_1) - g(\mathbf{p}_2)| \leq C||p_1 - p_2||_2 \tag{3.55}$$

Then the difference between revenues can be bounded:

$$|R - R_K| \leq \sum_{l=1}^{K} \sum_{i \in S_l} |g(\mathbf{p}_i) - g(\overline{\mathbf{p}}_l)| \leq C \sum_{l=1}^{K} \sum_{i \in S_l} ||\mathbf{p_i} - \overline{\mathbf{p}}_l||^2 \tag{3.56}$$

The last expression is the minimization objective of the K-means clustering (within cluster sum of squares) multiplied by a constant C. It supports our choice of the clustering algorithm. The revenue gap reduces as the clustering quality improves with at least linear rate. We verify this expression experimentally in the results section of this chapter.

If the pclick predictions are not accurate then the right hand side of the last expression will be perturbed by the estimation error ϵ. When the within cluster distance becomes smaller than ϵ, then $epsilon$ starts to dominate the upper bound and the improvement of the clustering quality will no longer guarantee the reduction of the revenue gap. As we mentioned earlier, this is an intuitive result, it does not make sense to consider the granularity with the resolution beyond the parameter estimation error.

Let us discuss the implications of the assumption, that g is Lipschitz continuous. This is a strong assumption; it actually states that the perturbation of the problem parameters does not lead to large changes in the objective. It is easy to show that GSP allocation rule satisfies this assumption when there are no budget constraints. If there are budget constraints, it is easy to come up with a counter example that breaks Lipschitz continuity. Consider two ads: a_1 has a large budget and a_2 has s small budget. The ad a_1 is ranked slightly higher than a_2, but pays significantly more per click. A small perturbation can flip the ranking of these two ads. This will result in high payments for the ad with small budget and low payments for the ad with large

budget. The same counter example can be used to show that the objective of our optimization based approach is not Lipschitz continuous. If we only have GD campaigns then under some mild regularity conditions we can use the result from [59] to prove that the objective is locally (but not globally) Lipschitz continuous. As we can see, Lipschitz continuity is hard to establish, but in real and very large problems the situations like the counter example above will have local effects. It is natural to assume that the objective function is well behaved on average. In the results section we will evaluate the convergence empirically to support this claim.

3.6 Results

We use the dataset from the previous chapter for evaluation. It contains 1000 queries and 1,5 million query submissions, i.e. 1500 submissions per query on average. The most frequent query "microsoft" was submitted about 300,000 times and the least frequent - 370 times. For each query a number of candidate ads is selected by the search engine during the preliminary filtering. The pclick estimations for these ads are available in the log and we use them as the features for clustering. The number of clusters per query ranges from 2 to 512. I.e. the number of left hand side nodes in the allocation graph can grow up to 512,000.[1] We limit the number of clusters to 128 for the flat allocation algorithm because the execution times get too long.

We test four different methods: pure auctions baseline, flat optimization and two variants of hierarchical optimization based on gradient projection and Dantzig-Wolfe decomposition. We use timing in addition to the evaluation metrics used in the previous chapter (revenue, clicks, cost per click and delivery rate). We use a computer cluster with 2.3Ghz 64bit nodes (AMD Opteron processors). Each allocation task is executed in a single thread without parallelization, i.e. one task per node at a time.

First, we analyze the quality of clustering using the within-cluster sum of squares (WCSS):

$$WCSS = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\| \tag{3.57}$$

where $\boldsymbol{\mu}_i$ is the centroid of the cluster S_i. The WCSS against the number of clusters is plotted in Figure 3.7. Quality improves very fast as the number of clusters per query grows. WCSS at K=512 amounts to 2.6% of WCSS at K=1. In section 3.5 we hypothesized that WCSS can be used to estimate the loss in the revenue and the number of clicks compared to the perfect clustering (zero WCSS) when we use one cluster per query submission. According to the hypothesis the loss is bounded by WCSS multiplied by a constant, equation (3.56). On figure 3.8 we report

[1] In reality the maximum number of nodes is slightly less because some queries have fewer than 512 distinct submissions.

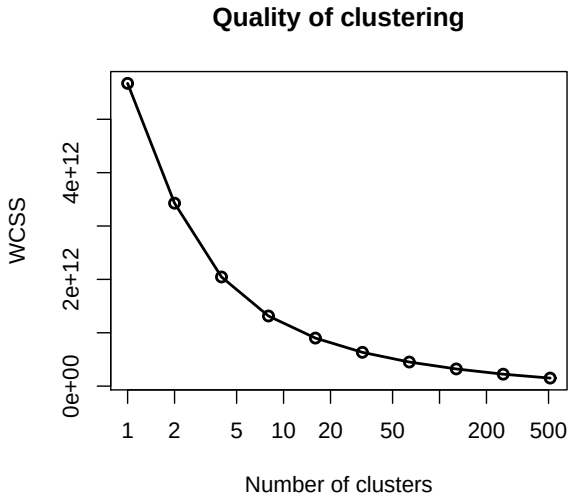


Figure 3.7: Quality of clustering: within cluster sum of squares.

the revenue and the number of clicks achieved by flat optimization as a function of WCSS. Both dependencies are almost linear. We conclude that the hypothesis is reasonable: WCSS is a good indicator of the revenue and the number of clicks. We can predict the perfect performance at $WCSS = 0$ that we could achieve with infinite computational resources. The predicted revenue at $WCSS = 0$ is around $2.04 \cdot 10^7$ which is only 0.7% better than the best revenue achieved by the flat optimization at K=128. The predicted number of clicks at $WCSS = 0$ is around $5.35 \cdot 10^5$ which is 2% better than the best value achieved by the flat optimization at K=128.

The revenue, clicks, cost per click, delivery rate and CPU time (in seconds) of different methods are reported on figure 3.9. All the metrics (except the timing) improve as the number of cluster increases. Flat optimization is the best in revenue and the number of clicks. Hierarchical optimization with gradient projection is slightly worse (around 0.6% in revenue and around 2% in clicks) than flat optimization, but it is much more efficient in terms of CPU time. For instance, for $K = 128$ flat optimization takes 13 hours while hierarchical optimization takes only 6 minutes: this is 2 orders of magnitude improvement. Figure 3.10 shows how fast each of the methods achieves certain performance.

Most of the results agree with our expectations and with the results of the previous chapter. For instance, our framework significantly outperforms the auction baseline in the number of clicks for all granularity levels. The relative improvement in the number of clicks increases from 9.5% for K=1 to 15.4% for K=512. The revenue of our framework is practically the same as the revenue achieved by the baseline. We already discussed this disproportion of the clicks and

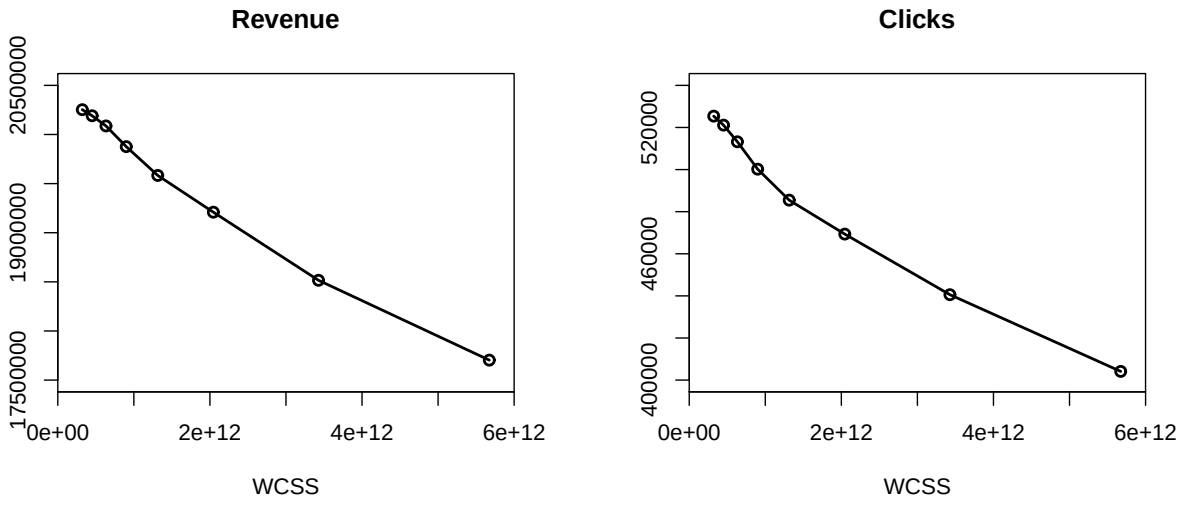


Figure 3.8: Revenue and the number of clicks of flat optimization as a function of clustering quality (WCSS). The measurements correspond to the numbers of clusters K=128, 64, 32, 16, 8, 4, 2, 1 (from left to right).

the revenue in the previous chapter. The search engine will be able to control the balance by the premium CPC payments. One unexpected result is that the hierarchical optimization with Dantzig-Wolfe decomposition has a surprisingly low revenue compared to the gradient decent. In our experience this method is quite sensitive to the choice of the initial solution. We plan to explore the ways to improve its performance in our future work.

To summarize the experimental findings, hierarchical optimization wins in scalability at the cost of objective value. The improvement in CPU time is very significant: two orders of magnitude and keeps increasing as the size of the problem increases. The loss in revenue and the number of clicks is quite small: 0.6% for the revenue and 2% for the clicks.

3.7 Summary

In this chapter we presented a hierarchical optimization algorithm for our ad allocation optimization framework. Formulating the allocation problem at the right granularity level is very important to achieve good performance. Coarse granularity leads to poor performance. Fine granularity faces scalability issues. The proposed algorithm decomposes one huge problem into a set of smaller problems that depend on each other through a special set of connecting variables. Sub-problems can be solved independently given the connecting variables. This provides great

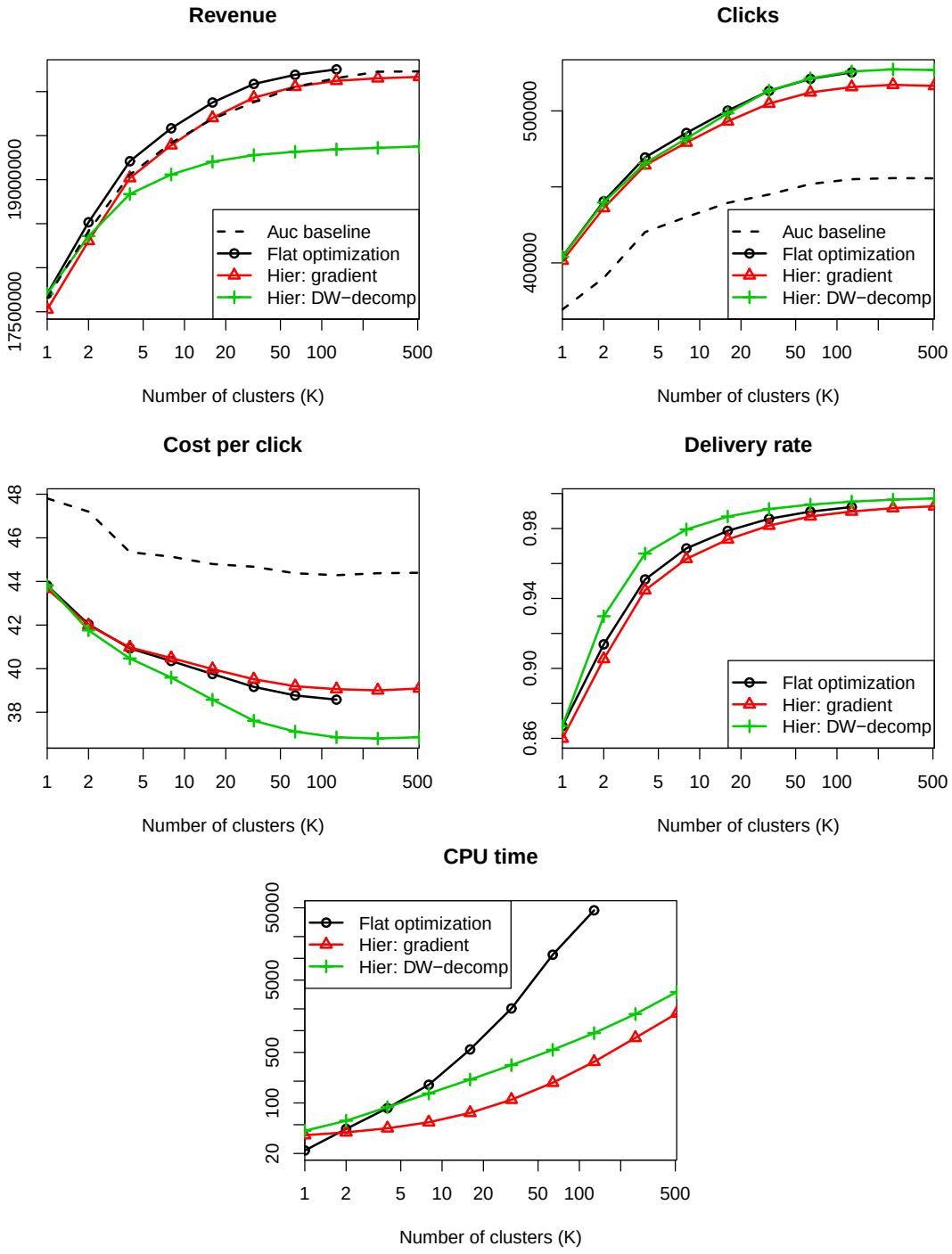


Figure 3.9: The main performance metrics as a function of the number of clusters per query, K. The largest K for flat optimization is 128. Pay attention to log scale of y-axis for CPU-time.

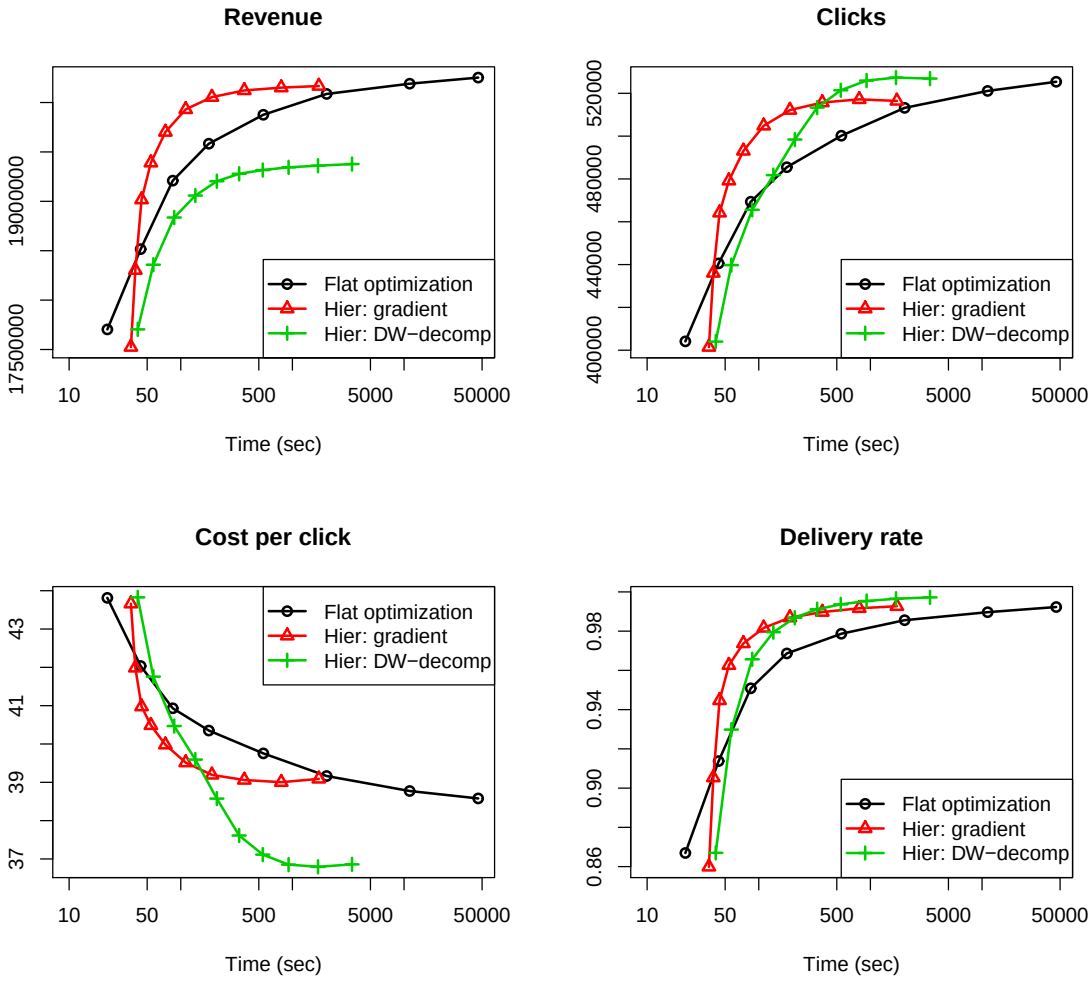


Figure 3.10: The main performance metrics as a function of CPU time.

potential for parallelization. The connecting variables are optimized by a dedicated master optimization algorithm. We show that the master optimization problem is convex. We propose two variants of master optimization: the gradient projection and Dantzig-Wolfe decomposition.

The experimental results confirm that hierarchical optimization provides great speed up (at least two orders of magnitude) at some cost of the objective value. In general, gradient descent methods are considered to be slow. The fact that we achieved good performance using gradient descent suggests that a better optimization algorithm that makes use of the problem structure is likely to exist. We hoped that Dantzig-Wolfe decomposition would be such a method, but it turned out to be inferior in terms of revenue. According to our observations Dantzig-Wolfe is very sensitive to the starting point. Finding a better starting point (e.g. by using a different optimization algorithm) can potentially improve the effectiveness. The main focus of our future work on hierarchical optimization is the reduction of the performance gap between flat and hierarchical optimization while maintaining the scalability of the latter.

The other direction of future work is problem specific graph partitioning and query clustering. In order to decompose the problem we partition the bipartite graph based on the query string. Other existing graph partitioning techniques, applied in machine learning (e.g. [74]) and graphical modeling, can potentially improve the overall quality of allocation. We will investigate the possibility of using a problem specific partitioning that depends on the objective function of the optimization framework. The same points apply to clustering of query submissions. The objective of the k-means clustering algorithm that we use in this work is not directly related to the optimization objective (although, the correlation exists as we show in section 3.5). We plan to design a problem specific clustering algorithm that depends on the framework objective (this idea was explored for display advertising in [68]).

Chapter 4

Game Theoretic Approach to Modeling of the Advertisers' Optimal Strategies

4.1 Introduction

In the previous two chapters we presented a novel ad allocation algorithm for sponsored search. Up to this point we investigated the problem from the position of the search engine. On our end the search engine gathers the information from all the advertisers and tries to satisfy their diverse needs and simultaneously maximizes its revenue. In this setup all the decisions of the advertisers are assumed to be given and fixed. In this chapter we change our perspective and focus on the advertisers: their reasoning and decision making. Our main interest is the optimal choice between auctions and guaranteed delivery, because the joint modeling of auctions and GD is the central point of our framework.

We say that we approach the problem from the advertiser's perspective, yet the ability to predict the advertisers' behavior is primarily beneficial for the search engine. For instance, it allows us to perform a more reliable evaluation of the algorithm. In chapter 2 we randomized the behavior of the advertisers for evaluation. Figure 4.1 depicts the average revenue and number of clicks from chapter 2 along with two extreme instances handpicked from the sample. Without the information about the actual decisions the gap that is spanned by these two extreme points demonstrates our uncertainty in the performance. In case of the revenue we are not even sure if the performance is above or below the baseline. The other benefit is that with a good model of the advertisers' behavior the search engine can identify the advertisers who make suboptimal decisions that harm both their own and the overall performances. The search engine can suggest such advertisers the optimal strategy and hence maximize the revenue (and other performance metrics). Clearly, the search engine is in the best position to optimize the performances of the advertisers because it possesses the information about all the participants, unlike the advertisers

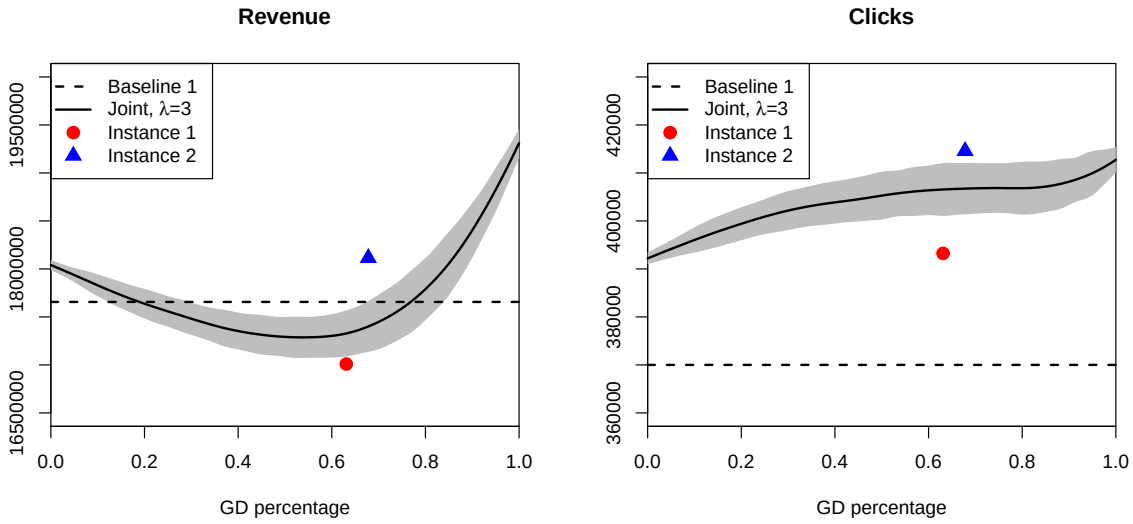


Figure 4.1: Average revenue and the number of clicks curves (from chapter 2). Instances 1 and 2 represent two realizations of advertisers decisions.

themselves or the third-party companies that manage the campaigns for the advertisers. Surprisingly, the search engine has an incentive to do so because it also improves its own revenue, as our results suggest.

To predict thr advertiser's behavior we have to solve a very sophisticated problem of interaction of multiple competing participants (players). Each advertiser has her own utility (we use the number of received clicks). In order to maximize this utility the advertisers adjust their bidding strategies (which words to bid on and how much) and their investment strategies based on the market. The investment strategies cover a set of long term decisions: the budget, the number of clicks and choice of either auction model or a guaranteed delivery model. What the optimal strategy is for each advertiser would depend on the strategies of all the other advertisers, and a change in the strategy by one advertiser would cause the changes of many other advertisers. Furthermore, such changes would unavoidably cause the online-ad engine to adjust its allocation of ads accordingly, which in turn changes the advertisers utilities and causes new adjustments in their investment strategies. Modeling of the actions of advertisers in such a dynamic environment is a game theoretic problem, where Nash equilibrium [53] plays a central role. Nash equilibrium characterizes the stationary and locally optimal state of the strategic interaction among multiple players. Informally Nash equilibrium corresponds to a set of decisions (also called strategies) of the players when no player can improve his utility by changing his decision given that the other players do not change theirs. Finding the Nash equilibrium of a game, if successful, allows bet-

ter understanding about the game and meaningful predictions such as the revenues of the search engine and the individual advertisers at the stationary point.

In this chapter we present the game theoretic analysis of the advertisers' decision between auctions and GD in the framework, presented in Chapter 2. We discover the Nash equilibrium of a game where the advertisers are allowed to switch between these two strategies. Among different existing variants of the equilibrium: pure strategy equilibrium, mixed strategy equilibrium and ϵ-approximate equilibrium, we focus our attention on the last one. Furthermore, we extend the standard definition of ϵ-approximation in a way that is the most appropriate for our setup. We explain and justify our choice in section 4.2.

Finding Nash equilibrium is generally a hard problem [21]. In online ads, equilibria has been studied for relatively simple scenarios, such as the bidding strategies of advertisers in single-shot auctions with the Generalized Second Price (GSP) rule [25, 26]. How to find Nash equilibrium in more complex cases, such as strategic switching between GD and auction in response to the changes of ad allocation by the search engine, has not been studied until this work. The main difficulty in the analysis is that the utilities of the advertisers cannot be represented as closed-form functions of the advertisers' strategies. Usually having well-defined utility functions is necessary (but not sufficient) for the analysis of the Nash equilibrium. If ad allocation relies on the optimization algorithm then the utilities are determined procedurally as the output of the optimization routine.

To address the above difficulties, we propose a novel approach to finding the Nash Equilibrium via local linear approximations of the utility functions. The key idea is to use ad-allocation algorithm to collect training examples for the regression (we use lasso regression [65]). The features of the regression are a vector of 0-1 values for the investment decisions (between auction and GD) by individual advertisers. The output is the click counts of the advertisers that we use as their utility functions. Having such a regression model trained on a sufficient number of input vectors, we obtain an approximation of the utility function for each advertiser. We use a combination of simulated annealing and integer linear programming (ILP) optimization to find the ϵ-approximate equilibrium for a set of linear utilities. In a two-stage algorithm the simulated annealing in the first stage finds the approximate solution and ILP in the second stage refines the approximation.

The chapter is organized as follows. In section 4.2 we discuss Nash equilibrium, its variants and outline the existing work on its computation. In section 4.3 we describe our framework and perform its theoretical analysis. Section 4.4 reports our experiments and results. We conclude the work and the future research in Section 4.5.

4.2 Nash Equilibrium

Nash equilibrium is the key concept in game theory. It is commonly used to analyze the strategic interaction of multiple players. First, let us define such interaction, also called a game. The following tuple is called a game: a set of players $i = 1..n$, a set of strategies or moves $z_i \in Z_i$ available to each player i and a real-valued utility functions u_i of each player i for each possible outcome of the game[1]. A vector of strategies/moves selected by the players $\mathbf{z} = \{z_1 \dots z_n\}$ is called a strategic profile. The outcome of the game in our scenario is a deterministic function of the strategic profile:

$$\mathcal{Q}(z_1, \dots z_n) \tag{4.1}$$

In the context of our ad allocation problem, $\mathcal{Q}$ is simply the solution of our optimization problem for a particular input from the advertisers. A real-valued utility function u_i is a function of the outcome and is therefore a function of a strategic profile:

$$u_i = u_i(\mathcal{Q}(z_1, \dots z_n)) = u_i(z_1, \dots, z_n) \tag{4.2}$$

We use the expected number of clicks as a utility function. This quantity can be easily computed given the ad allocation $\mathcal{Q}$. The players select their moves z_i simultaneously and do not know the moves of the opponents.

To predict the outcome of the game, the game theorists introduce the concept of the game equilibrium: a state of the game when no player wants to change his/her strategy. There exist different variants of the equilibrium: pure strategy Nash equilibrium (introduced as early as 1838 by A. Cournot [18]), mixed strategy Nash equilibrium [51], ϵ-approximate equilibrium [52, p.45] etc. These definitions have different interpretation and properties. The choice of the right one as the analysis tool depends on the nature of the game. We outline each of these equilibrium definitions below, discuss their applicability to our problem and make the choice.

4.2.1 Pure Strategy Nash Equilibrium

A strategic profic $\mathbf{z}^*$ is a Nash equilibrium in pure strategies if:

$$\forall i, z_i \in Z_i : \quad u_i(z_i^*, \mathbf{z}_{-i}^*) \geq u_i(z_i, \mathbf{z}_{-i}^*) \tag{4.3}$$

where $\mathbf{z}_{-i}$ denotes the decisions of all the players except the decision of the player i:

$$\mathbf{z}_{-i} = \{z_1 \dots z_{i-1}, z_{i+1} \dots z_n\} \tag{4.4}$$

[1] A standard definition of a game has a preference relation for every pair of outcomes instead of a utility function. Using a real valued utility function is one way to define such preference relation.

In other words in the equilibrium state no player can improve his utility by changing his strategy unilaterally, i.e. when the other players do not change theirs. Not every game has Nash equilibrium in pure strategies and some games may have more than one. Specifically, the game that we are investigating may not have the equilibrium. We will demonstrate it using a counter example later in the chapter. In case of multiple equilibria, finding the one with the worst performance (social welfare, the sum of players utilities) is of the particular theoretical interest.

4.2.2 Mixed Strategy Nash Equilibrium

Many games do not have Nash equilibrium in pure strategies. This makes N.E. apparatus useless for analysis of such games. In order to overcome this problem the mixed strategies were introduced. Essentially, instead of choosing one of the moves $z_i \in Z_i$ deterministically, the player chooses the actions probabilistically by assigning probabilities to each move:

$$p_i(z) : \sum_{z \in Z_i} p_i(z) = 1 \tag{4.5}$$

The utility functions are computed as the expectations of the utilities of pure strategies:

$$u_i'(p_1 \dots p_n) = E[u_i(z_1 \dots z_n)] = \sum_{z_1 \dots z_n} \prod_{i=1}^{n} p_i(z_i) u_i(z_1 \dots z_n) \tag{4.6}$$

Nash equilibrium in mixed strategies is a set of probability vectors $p_1^* \dots p_n^*$ that satisfy:

$$\forall i, p_i \ \ (||p_i|| = 1) : \quad u_i'(p_i^*, p_{-i}^*) \geq u_i'(p_i, p_{-i}^*) \tag{4.7}$$

If the number of the players and the number of pure strategies Z_i are finite then there exists at least one Nash equilibrium in mixed strategies. This result, proved by Nash in [51], follows from Kakutani's fixed point theorem [37]. The main criticism of the mixed strategies is that the random behavior is not typical for humans.

4.2.3 ϵ-approximate equilibrium

The approximate Nash equilibrium relaxes the original conditions of the Nash equilibrium (4.3). Now each player can have a small incentive to change his strategy :

$$\forall i, z_i \in Z_i : \quad u_i(z_i^*, \mathbf{z}_{-i}^*) + \epsilon \geq u_i(z_i, \mathbf{z}_{-i}^*) \tag{4.8}$$

In other words each player can win at most ϵ if he changes his strategy. The approximate equilibrium can be defined for both pure and mixed strategy equilibria. In the definition above we use pure strategies. Zero value of ϵ corresponds to the exact Nash equilibrium.

Approximate equilibrium is more attractive than the Nash equilibrium for computational reasons. It is easier to compute and is more natural when approximate computational algorithms are used [52]. In addition, the games that do not have an equilibrium in pure strategies still have an approximate equilibrium for some value of ϵ.

4.2.4 Our Choice

From the discussion it is clear that the ϵ-approximate equilibrium is more appropriate for our setup. Our framework has several sources of approximation errors: click prediction, supply forecast and the utility function approximation. When it comes to equilibrium computation we do not gain anything by finding the exact equilibrium of a game that has errors in its specification. We also need to choose between pure strategies and mixed strategies. We believe that the former concept represents a more realistic scenario. It is quite unlikely that the advertisers make the strategic decisions randomly. Yet our approach is capable of handling the mixed strategies too.

We propose two modifications of the approximate equilibrium (4.8). According to the definition above the tightness of the approximation is determined by the unhappiest player:

$$\Delta u_i = \max_{z_i \in Z_i} (u_i(z_i, \mathbf{z}^*_{-i}) - u_i(z^*_i, \mathbf{z}^*_{-i})) \tag{4.9}$$

$$\epsilon_{tight} = \max_i \Delta u_i \tag{4.10}$$

Online advertising involves thousands of players and statistically the probability that at least one of them is very unhappy is very high. The maximum and the minimum are the least robust summary statistics of a sample. Also, we have very diverse advertisers whose budgets may differ by several orders of magnitude. Under such circumstances relying on the performance of a potential outlier may not be representative enough. Instead, we propose two alternative ways to define ϵ. In the first definition we count the percentage of the advertisers who can improve their performances by changing the strategy. This way we ignore the size of the advertisers:

$$\epsilon = \frac{1}{n} \sum_{i=1}^{n} I(\Delta u_i > 0) \tag{4.11}$$

Where I is an indicator function ($I(true) = 1$ and $I(false) = 0$). The second way is to sum all the unhappy advertisers instead of finding the unhappiest one:

$$\epsilon = \sum_i \Delta u_i \tag{4.12}$$

4.2.5 Computation of Nash Equilibrium

Numeric computation of Nash Equilibrium is considered a hard problem (not solvable in polynomial time) though no formal proof exists [21] [52]. Most research focuses on finding mixed strategy equilibrium when the players are allowed to assign probabilities to their strategies and therefore randomize their behaviors. Unlike pure strategy equilibrium that may or may not exist, at least one mixed equilibrium always exists [51].

For a small set of games Nash Equilibrium can be found analytically by solving the system of utility maximization conditions. Second prize auction is an example of such a game. When the analytical solution is not possible (most real games) the numerical methods are applied. Famous algorithms are Lemke-Howson algorithm [45] for 2-player games and simplicial subdivision based algorithms (e.g. [66]) for n-player games. These algorithms assume many strategies per player and have exponential worst case performance. The exponential complexity is a result of combinatorially many subsets of pure strategies that form mixed strategies. Our problem is conceptually different: each player has only 2 pure strategies, but there are thousands of players. The space requirement to store the utility functions of N players is $N2^N$. The computation of one value of the utility function can take significant time. This makes many popular approaches useless for our problem.

A totally different approach to equilibrium computation is based on reinforcement learning. In the reinforcement learning setup an agent/player tries to maximize his rewards in a multi-step game. The player chooses one action per move. The immediate rewards for choosing an action are uncertain (e.g. due to unknown actions of the opponents) and the player learns these uncertainties in the course of the game by randomizing his behavior. The likelihood to choose the action depends on the past rewards associated with this action. In game theory reinforcement learning techniques, motivated by human behavior, are used to explain how the real systems arrive to equilibrium states. Some authors go in the reverse direction and simulate the learning process to discover the equilibrium. For instance, in [50] the exploration/exploitation algorithm is empirically shown to converge to the Nash equilibrium for a set of benchmark games. According to this algorithm at each step the player either exploits by choosing the best action or explores by choosing a random alternative action. The probability of exploration diminishes with time. The algorithm does not require computation of the complete utility function matrix and therefore can be used in our setup (we use it as a baseline). The choice of the right learning parameters (e.g. the decay rate of the exploration probability) is very important. The algorithm may require a lot of iterations to converge which is critical if the call to evaluate the utility function is time-costly. The algorithm is sequential and therefore hard to parallelize.

The algorithm proposed in this work can be seen as a mixture of analytic and simulation approaches. We use the learning techniques to systematically explore the strategic space of the game. Then we use theoretically appealing methods to leverage the information obtained through

learning and to find the equilibrium.

4.3 Approximate Nash Equilibrium for Auctions and GD in Sponsored Search

In this section we formulate the problem of choice between auctions and guaranteed delivery as a game theoretic problem of finding equilibrium strategies. We use the expected number of clicks as advertiser's utility. The original utility function cannot be expressed in closed form and is not amenable to analysis. To overcome this difficulty we use local linear approximations of the utility functions. In order to find Nash equilibrium of a game with linear utilities we formulate several optimization problems. In sections 4.3.1 and 4.3.2 we describe the utility approximation algorithm and provide theoretic assessment of our approach, i.e. how the utility function approximation affects the accuracy of the equilibrium. In section 4.3.3 we solve the optimization problems to find the equilibrium. Two of our optimization problems aim to find the tightest ϵ-equilibrium (with smallest ϵ) for two different definitions of ϵ, given in (4.11) and (4.12). The other two optimization problems find the worst case social welfare for a fixed level of approximation (fixed value of ϵ).

In section 2.4 we formulated the ad allocation as an optimization problem, formula (2.14) presented below for convenience:

$$\begin{aligned} \max_{\mathbf{x},\xi_c} \; & \boldsymbol{\alpha}^\intercal \mathbf{x} + \sum_{c \in GD} (d_c - \mu_c \xi_c) \\ \text{s.t.} \quad & \boldsymbol{\delta}_c^\intercal \mathbf{x} \le d_c \quad \forall c \in A \\ & \boldsymbol{\beta}_c^\intercal \mathbf{x} - \xi_c \le -m_c \quad \forall c \in GD \\ & \mathbf{1}^\intercal \mathbf{x}_i \le 1 \quad \forall i \\ & \mathbf{x}, \xi_c \ge 0 \end{aligned} \tag{4.13}$$

The expected number of clicks for a given solution $\mathbf{x}$ for any campaign is:

$$u_i = -\boldsymbol{\beta}_i^\intercal \mathbf{x} \quad \forall i \in A \cup GD \tag{4.14}$$

The solution $\mathbf{x}$ depends on the numerous parameters of the optimization routine that include pclick predictions, information about queries, advertisers' bids, budgets, click requirements and their choice between auctions and guaranteed delivery. We focus our attention on the last parameter, the choice between auctions and GD for every advertiser. For convenience we denote this complex dependency between the input parameters and the solution of the LP as $\mathcal{Q}$:

$$\mathbf{x} = \mathcal{Q}(z_1 \dots z_n, \eta) \tag{4.15}$$

The binary variable $z_i \in \{0, 1\}$ expresses the choice of the advertiser between auctions ($z_i = 0$) and GD ($z_i = 1$). The variable η covers all other parameters. The utility of the advertiser is then:

$$u_i = -\boldsymbol{\beta}_i^{\mathsf{T}} \mathcal{Q}(z_1 ... z_n, \eta) = f_i(z_1 ... z_n) \tag{4.16}$$

Each advertiser is only allowed two choices. It is convenient to distinguish the utility functions for either of these choices:

$$u_{i,0} = f_i(\mathbf{z}|z_i = 0) = f_{i,0}(\mathbf{z}) \tag{4.17}$$

$$u_{i,1} = f_i(\mathbf{z}|z_i = 1) = f_{i,1}(\mathbf{z}) \tag{4.18}$$

Let us define Δu_i as the amount that the advertiser can gain or lose if he changes his strategy from z_i to $1 - z_i$:

$$\Delta u_i = f_{i,1-z_i}(\mathbf{z}) - f_{i,z_i}(\mathbf{z}) \tag{4.19}$$

Then finding the tightest ϵ-equilibrium for two definitions of ϵ can be formulated as:

$$\text{OPT1}: \quad \min_{\mathbf{z}} \sum_{i=1}^{n} I(\Delta u_i(\mathbf{z}) > 0) \tag{4.20}$$

$$\text{OPT2}: \quad \min_{\mathbf{z}} \sum_{i=1}^{n} \Delta u_i(\mathbf{z}) I(\Delta u_i(\mathbf{z}) > 0) \tag{4.21}$$

OPT1 finds the state with the smallest number of unhappy advertisers, i.e. minimizes ϵ from (4.11). OPT2 finds the state with the smallest sum of losses of unhappy advertisers, i.e. minimizes ϵ from (4.12).

The worst case analysis for the social welfare for a given value of ϵ is then formulated as:

$$\text{OPT3}: \quad \min_{\mathbf{z}} \sum_{i=1}^{n} f_i(\mathbf{z}) \quad \text{Subject to:} \quad \sum_{i=1}^{n} I(\Delta u_i(\mathbf{z}) > 0) \leq \epsilon \tag{4.22}$$

$$\text{OPT4}: \quad \min_{\mathbf{z}} \sum_{i=1}^{n} f_i(\mathbf{z}) \quad \text{Subject to:} \quad \sum_{i=1}^{n} \Delta u_i(\mathbf{z}) I(\Delta u_i(\mathbf{z}) > 0) \leq \epsilon \tag{4.23}$$

The objective of these two problems is simply the sum of the utilities of the advertisers.

4.3.1 Approximate Utility and Convergence Analysis

Unfortunately, the solution of a linear program $\mathcal{Q}$ (and consequently the utilities $f_{i,.}$) cannot be expressed in closed form as a function of advertisers' decisions. To address this problem we propose to replace f in the problems OPT1-4 with its approximation $\tilde{f}$:

$$f_i(\mathbf{z}) \approx \tilde{f}_i(\mathbf{z}) \tag{4.24}$$

We will use the tilde $\tilde{}$ to denote any expression that is obtained by replacing the actual utility functions with their approximations. For example,

$$\Delta\tilde{u}_i = \tilde{f}_{i,1-z_i}(\mathbf{z}) - \tilde{f}_{i,z_i}(\mathbf{z}) \tag{4.25}$$

The approximate versions of OPT1-OPT4 are the optimization problems obtained from OPT1-OPT4 by replacing the utility functions with their approximations. To justify this substitution we need to show that a good approximation of the utility leads to a good approximation of the problem solution. This property can be proven for the problems OPT2 and OPT4, but not for OPT1 and OPT3. We formulate it as the following theorems.

Theorem 4.3.1 *Let $\tilde{f}_i(\mathbf{z})$ be a uniform approximation of $f_i(\mathbf{z})$ for all players, i.e. $\forall i, \mathbf{z} : |\tilde{f}_i(\mathbf{z}) - f_i(\mathbf{z})| \leq \delta_i$. Then the difference between the true minimum and the minimum of the approximate OPT2* (4.21) *is at most $2\sum_i \delta_i$.*

Informally, the theorem states that if the approximations of the utility functions are "good", then the approximation of the objective function is also "good" (we prove this statement below). Since two functions are close to each other, it follows that the minimum value of the former one is close to the minimum value of the latter.

Proof First, $\Delta\tilde{u}_i$ uniformly approximates Δu_i:

$$|\Delta\tilde{u}_i - \Delta u_i| = |f_{i,0} - f_{i,1} - \tilde{f}_{i,0} + \tilde{f}_{i,1}| \leq |f_{i,0} - \tilde{f}_{i,0}| + |f_{i,1} - \tilde{f}_{i,1}| \leq 2\delta_i \tag{4.26}$$

Now we show that each summand in the objective of the approximate OPT2, $\Delta\tilde{u}_i I(\Delta\tilde{u}_i > 0)$, approximates the summand in the objective of the original OPT2, $\Delta u_i I(\Delta u_i > 0)$. Consider two cases.

Case 1: Δu_i and $\Delta\tilde{u}_i$ have the same sign.

$$I(\Delta u_i > 0) = I(\Delta\tilde{u}_i > 0)$$
$$|\Delta u_i I(\Delta u_i > 0) - \Delta\tilde{u}_i I(\Delta\tilde{u}_i > 0)| = |\Delta u_i - \Delta\tilde{u}_i|\, I(\Delta u_i > 0) \leq |\Delta\tilde{u}_i - \Delta u_i| \leq 2\delta_i$$

Case 2: Δu_i and $\Delta\tilde{u}_i$ have different signs.

$$\begin{aligned}
&|\Delta u_i I(\Delta u_i > 0) - \Delta\tilde{u}_i I(\Delta\tilde{u}_i > 0)| \\
&\leq |\Delta u_i| I(\Delta u_i > 0) + |\Delta\tilde{u}_i| I(\Delta\tilde{u}_i > 0) \\
&\leq |\Delta\tilde{u}_i| + |\Delta u_i| = |\Delta\tilde{u}_i - \Delta u_i| \leq 2\delta_i
\end{aligned}$$

Both cases give the same upper bound $2\delta_i$. Finally, we can establish the approximation for the objective function of OPT2 as the sum of approximation factors of individual summands, i.e. $2\sum_i \delta_i$. ∎

Theorem 4.3.2 *Let $\tilde{f}_i(\mathbf{z})$ be a uniform approximation of $f_i(\mathbf{z})$ for all players, i.e. $\forall i, \mathbf{z} : |\tilde{f}_i(\mathbf{z}) - f_i(\mathbf{z})| \leq \delta_i$. Let $g(\epsilon)$ be the minimum of the original OPT4 and $\tilde{g}(\epsilon)$ be the minimum of the approximate OPT4. Let $\tau = \sum_i \delta_i$ Then the following inequality holds:*

$$\tilde{g}(\epsilon + 2\tau) - \tau \leq g(\epsilon) \leq \tilde{g}(\epsilon - 2\tau) + \tau \tag{4.27}$$

The statement of this theorem might not be intuitive because we deal with constrained optimization problem. Similar to the first theorem, we show that the objective function and the approximate objective are close to each other. But the feasibility regions may differ, therefore we need an extra step that establishes the relationship between the feasibility regions of the original optimization and the approximate optimization. We use the monotonicity of the feasibility region as a function of ϵ, i.e. as ϵ increases, the feasibility region does not decrease. Then we establish the following relationship between the feasibility regions:

$$FeasApprox(\epsilon_1) \subseteq FeasOrig(\epsilon) \subseteq FeasApprox(\epsilon_2) \tag{4.28}$$

for some values of ϵ, ϵ_1 and ϵ_2 that are close to each other. Finally, we use the fact that the minimum of the function monotonically decreases as the feasibility region grows.

Proof We prove the theorem in two steps that can be summarized as follows. First, we properly bound the feasibility region of OPT4. Second, we bound the objective function. Together these two bounds provide the bound on the minimum of the optimization. The feasibility region of OPT4 is:

$$\mathbf{z} \in Z(\epsilon) : \quad \sum_{i=1}^{n} \Delta u_i(\mathbf{z}) I(\Delta u_i(\mathbf{z}) > 0) \leq \epsilon \tag{4.29}$$

The left hand side of the inequality is the objective of OPT2 and the previous theorem already establishes the bounds on this expression in terms of utility approximation:

$$\sum_{i=1}^{n} \Delta\tilde{u}_i(\mathbf{z}) I(\Delta\tilde{u}_i(\mathbf{z}) > 0) - 2\tau \leq \sum_{i=1}^{n} \Delta u_i(\mathbf{z}) I(\Delta u_i(\mathbf{z}) > 0) \leq \sum_{i=1}^{n} \Delta\tilde{u}_i(\mathbf{z}) I(\Delta\tilde{u}_i(\mathbf{z}) > 0) + 2\tau$$

Let's take the lower bound and combine it with the inequality (4.29):

$$\sum_{i=1}^{n} \Delta\tilde{u}_i(\mathbf{z}) I(\Delta\tilde{u}_i(\mathbf{z}) > 0) - 2\tau \overset{\forall \mathbf{z}}{\leq} \sum_{i=1}^{n} \Delta u_i(\mathbf{z}) I(\Delta u_i(\mathbf{z}) > 0) \overset{\mathbf{z} \in Z(\epsilon)}{\leq} \epsilon$$

The first inequality holds for all $\mathbf{z}$ the second holds for $\mathbf{z} \in Z(\epsilon)$, therefore the following inequality holds:

$$\sum_{i=1}^{n} \Delta\tilde{u}_i(\mathbf{z}) I(\Delta\tilde{u}_i(\mathbf{z}) > 0) - 2\tau \overset{\mathbf{z} \in Z(\epsilon)}{\leq} \epsilon$$

And it actually expresses the feasibility region of the approximate OPT4, $\tilde{Z}(\epsilon + 2\tau) \supseteq Z(\epsilon)$. We get the lower bound of the feasiblity region in the same way, by using the upper bound of the constraint expression:

$$\tilde{Z}(\epsilon + 2\tau) \supseteq Z(\epsilon) \supseteq \tilde{Z}(\epsilon - 2\tau) \tag{4.30}$$

Now we need to combine the bounds for the feasibility region with the bounds on the objective function. These bounds follow directly from the first condition of the theorem:

$$\sum_{i=1}^{n} \tilde{f}_i(\mathbf{z}) - \tau \leq \sum_{i=1}^{n} f_i(\mathbf{z}) \leq \sum_{i=1}^{n} \tilde{f}_i(\mathbf{z}) + \tau \tag{4.31}$$

Together the bounds on the objective function and the bounds on the feasibility region give us the bounds on the minimum of the optimization:

$$\tilde{g}(\epsilon + 2\tau) - \tau \leq g(\epsilon) \leq \tilde{g}(\epsilon - 2\tau) + \tau \tag{4.32}$$

which completes the proof of the theorem. ∎

We cannot get the same style approximation bounds for the problems OPT1 and OPT3 because the count of unhappy advertisers is not a continuous function. If an advertiser has a very small difference in the utilities for auctions and GD, then even small error in utility approximation can cause the change of the strategy. To prove the tight approximation rate for OPT1 and OPT3 we need to request an extra property for the data. Namely, we need the difference between auction and GD utilities to be large: $\min_{i,\mathbf{z}} |\Delta u_i(\mathbf{z})| \geq C > 0$. If the approximation of the utilities is worse than C then we cannot guarantee tight bounds, if the approximation is better than C then we immediately get the exact solution. For practical use this bound is not very useful because C can be very small or even zero. We may never get such a good approximation of the objective function. Still, even a method that does not guarantee a tight bound can be useful in practice therefore we do not disregard OPT1 and OPT3.

4.3.2 Learning the Utility Function

The next question is: how to choose the approximation $\tilde{f}$? There are two necessary conditions: 1) the approximation should be accurate and 2) we should be able to solve OPT1-4 for the chosen approximation. Our choice is a local linear approximation. We train two functions to approximate each advertiser's utility: one for the case when he selects auctions and the other one for the case when he selects GD. Our main goal is to model the difference between these two

cases, $\Delta\tilde{f}_i$:

$$\tilde{f}_{i,0}(\mathbf{z}) = \mathbf{a}_{i,0}^{\intercal}\mathbf{z} + b_{i,0} \tag{4.33}$$

$$\tilde{f}_{i,1}(\mathbf{z}) = \mathbf{a}_{i,1}^{\intercal}\mathbf{z} + b_{i,1} \tag{4.34}$$

$$\Delta\tilde{f}_i(\mathbf{z}) = \tilde{f}_{i,1} - \tilde{f}_{i,0} = \mathbf{a}_i^{\intercal}\mathbf{z} + b_i \tag{4.35}$$

Without loss of generality we set $a_{ii,0} = a_{ii,1} = 0$ because the decision of the i-th player is fixed in $\tilde{f}_{i,0/1}$.

We need to fit $2n$ linear functions (2 per advertiser) and the number of features in each function is $n - 1$. To fit the parameters $\mathbf{a}$ and b we generate the training data by running the allocation algorithm $\mathcal{Q}(\mathbf{z})$ for randomly generated inputs $\mathbf{z}$. One execution of the allocation algorithm gives us one training instance for each advertiser. The dimensionality of the feature space is large and the sampled data will inevitably be sparse (the problem is known as *the curse of dimensionality*). The right choice of the sampling distribution and the use of the active learning techniques [61] can address this problem to a degree. We use the following approach: we start sampling uniformly, and periodically learn the approximations and solve the optimization for these approximations, and then we start sampling closer to the current solution. This approach allows us to get a more accurate estimate around the solution. We summarize this approach in algorithm 1.

4.3.3 Nash Equilibrium for Linear Utility Functions

Now we explain how to solve OPT1-4 ((4.20)-(4.23)) for the linear utility functions. We provide three different solutions: the simulated annealing [42], integer linear programming (ILP) and the hybrid approach. Each of these methods has its pros and cons. Simulated annealing is fast and simple, but it can get stuck in a local optimum and therefore requires multiple restarts. Also, it requires careful tuning of the parameters (cooling schedule). ILP is guaranteed to find the optimal solution but it is very slow. A generic ILP with binary variables is NP-complete [38]. The hybrid approach uses ILP to refine the solution obtained by the simulated annealing. This approach allows to trade off the accuracy for the execution time.

Simulated annealing. The simulated annealing works in the following way. At each step we select one advertiser and change his strategy. If the objective value improves we accept the change, if the objective value worsens, we accept the change with some probability and revert the change otherwise. The acceptance probability depends on the change of the objective (the higher the difference - the lower the probability) and the time (probability reduces with time). The algorithm is summarized in algorithm 2.

The problems OPT1 and OPT2 can be solved by this algorithm directly. The updates of the objective function can be computed very efficiently. After flipping a variable z_i we need to

Algorithm 1 Learning the utility approximation

Set $z_i^* = 0 \; \forall i$ - initial point
Set $p_{max} = 1$
while Repeat until $\mathbf{z}^*$ converges **do**
  for Sample M training points **do**
    select $p \sim \text{Uniform}(0, p_{max})$
    for $i = 1 \dots n$ **do**
      Set $z_i = z_i^*$ with probability $1 - p$ and $z_i = (1 - z_i^*)$ with probability p
    end for
    Add this point to the training data
  end for
  Re-train all the regression functions. We use lasso regression [65] that provides sparse solution and good accuracy. Lasso minimizes the sum of squared errors regularized by L1-norm:

$$\min_{\mathbf{a},b} \sum_j \left(f(\mathbf{z}_j) - \mathbf{a}^\intercal \mathbf{z}_j - b\right)^2 + \lambda \left(|b| + \|\mathbf{a}\|_1\right) \tag{4.36}$$

  here j is the summation over training instances.
  Solve the optimization (covered in section 4.3.3), update $\mathbf{z}^*$ with new solution.
end while

update the utilities of all the players that potentially depend on z_i, i.e. the players j with $a_{ji} \neq 0$. Therefore the complexity of the update is $O(n)$. The update is even faster if the matrix A is sparse.

The problems OPT3 and OPT4 are constrained optimization problems and cannot be solved by the algorithm 2 directly. We use the method of Lagrange multipliers to convert these problems to unconstrained optimization:

$$\text{OPT3'}: \quad \min_{\mathbf{z}} \sum_{i=1}^{n} f_i(\mathbf{z}) + \lambda \sum_{i=1}^{n} I(\Delta u_i(\mathbf{z}) > 0) \tag{4.37}$$

$$\text{OPT4'}: \quad \min_{\mathbf{z}} \sum_{i=1}^{n} f_i(\mathbf{z}) + \lambda \sum_{i=1}^{n} \Delta u_i(\mathbf{z}) I(\Delta u_i(\mathbf{z}) > 0) \tag{4.38}$$

These problems can be solved by simulated annealing for selected values of the Lagrange coefficient λ, i.e. the solution depends on λ: $\mathbf{z}^* = \mathbf{z}^*(\lambda)$. The solution of the original OPT3 and OPT4 depends on the parameter ϵ: $\mathbf{z}^* = \mathbf{z}^*(\epsilon)$. To convert the solution $\mathbf{z}^*(\lambda)$ into this form we use the

Algorithm 2 Simulated annealing, minimize $g(\mathbf{z})$

initialize $\mathbf{z}$ randomly
set solution $\mathbf{z}^* = \mathbf{z}$ and objective $e^* = g(\mathbf{z})$
select p_0 and T - cooling schedule parameters
for $time = 1..M$ **do**
  for $i = 1..N$ **do**
    Set $\mathbf{z}'$: $z'_i = 1 - z_i$ and $z'_j = z_j \ \ \forall j \neq i$
    if $g(\mathbf{z}') < g(\mathbf{z})$ set $\mathbf{z} = \mathbf{z}'$
    else set $\mathbf{z} = \mathbf{z}'$ with probability
    $p = p_0 \exp(-(g(\mathbf{z}') - g(\mathbf{z}))time/T)$
    if $g(\mathbf{z}) < e^*$ **then**
      $\mathbf{z}^* = \mathbf{z}$ and $e^* = g(\mathbf{z})$
    end if
  end for
end for

dependencies between λ and ϵ:

$$\epsilon(\lambda) = \sum_{i=1}^{n} I(\Delta u_i(\mathbf{z}^*(\lambda)) > 0) \qquad \text{for OPT3} \tag{4.39}$$

$$\epsilon(\lambda) = \sum_{i=1}^{n} \Delta u_i(\mathbf{z}^*(\lambda)) I(\Delta u_i(\mathbf{z}^*(\lambda)) > 0) \quad \text{for OPT4} \tag{4.40}$$

Integer linear programming. Another way to solve the problems OPT1-4 is to reformulate them as integer linear programs (ILP) with binary variables. A generic ILP with binary variables is NP-complete [38] and is usually solved by the branch and cut algorithm [48][2]. First, we encode the condition that the advertiser is happy as a set of linear inequalities. Recall that

$$\Delta \tilde{f}_i(\mathbf{z}) = \tilde{f}_{i,1} - \tilde{f}_{i,0} = \mathbf{a}_i^\intercal \mathbf{z} + b_i \tag{4.41}$$

is the difference between the GD utility and the auction utility of the advertiser. The advertiser is happy if either of these conditions is satisfied:

$$\left[\begin{array}{l} \mathbf{a}_i^\intercal \mathbf{z} + b_i \geq 0, \ z_i = 1 \\ \mathbf{a}_i^\intercal \mathbf{z} + b_i \leq 0, \ z_i = 0 \end{array} \right. \tag{4.42}$$

[2]We can make use of the structure of the problems and improve the computation of the bounds in the branch and cut algorithm. This does not affect the class of the complexity but can improve its constants.

The first condition states that the advertiser chose GD and his GD utility is not worse than his auction utility. The second condition states that the advertiser chose auctions and his auction utility is not worse than his GD utility. These conditions look almost like a set of the constraints for an integer linear program. However, the constraints of ILP should be joined by AND operator while our constraints are joined by OR operator. An algebraic trick allows us to reformulate OR into AND:

$$\left[\begin{array}{l}\mathbf{a}_i^\intercal\mathbf{z}+b_i\geq 0, z_i=1\\ \mathbf{a}_i^\intercal\mathbf{z}+b_i\leq 0, z_i=0\end{array}\right. \iff \left\{\begin{array}{l}\mathbf{a}_i^\intercal\mathbf{z}+b_i+m_i(1-z_i)\geq 0\\ \mathbf{a}_i^\intercal\mathbf{z}+b_i-M_iz_i\leq 0\\ z_i\in\{0,1\}\end{array}\right. \tag{4.43}$$

where m_i and M_i are large constants that dominate the value of the constraint and guarantee its satisfaction unless multiplied by zero:

$$M_i\geq\max_{\mathbf{z}}(\mathbf{a}_i^\intercal\mathbf{z}+b_i) \tag{4.44}$$

$$m_i\geq\min_{\mathbf{z}}(\mathbf{a}_i^\intercal\mathbf{z}+b_i) \tag{4.45}$$

Equivalence in (4.43) is verified by substitution of $z_i=1$ and $z_i=0$. If we combine the happiness conditions for all the players, we get the feasibility problem for the pure strategy Nash equilibrium:

$$\begin{aligned}\mathbf{Az}+\mathbf{b}+\mathbf{m}^\intercal\mathbf{I}(\bar{1}-\mathbf{z})&\geq 0\\ \mathbf{Az}+\mathbf{b}-\mathbf{M}^\intercal\mathbf{Iz}&\leq 0\end{aligned} \tag{4.46}$$

where $\mathbf{I}$ is the identity matrix. To allow the advertisers to be unhappy we introduce the binary slack variables: $y_i=0$ if the player i is happy and $y_i=1$ otherwise:

$$\mathbf{Az}+\mathbf{b}+\mathbf{m}^\intercal\mathbf{I}(\bar{1}-\mathbf{z}+\mathbf{y})\geq 0 \tag{4.47}$$

$$\mathbf{Az}+\mathbf{b}-\mathbf{M}^\intercal\mathbf{I}(\mathbf{z}+\mathbf{y})\leq 0 \tag{4.48}$$

Now it is easy to formulate OPT1 that simply minimizes the number of unhappy advertisers as an ILP:

$$\begin{array}{l}\min_{\mathbf{z},\mathbf{y}}\ \bar{1}^\intercal\mathbf{y}\\ \text{s.t.}\quad \mathbf{Az}+\mathbf{b}+\mathbf{m}^\intercal\mathbf{I}(\bar{1}-\mathbf{z}+\mathbf{y})\geq 0\\ \mathbf{Az}+\mathbf{b}-\mathbf{M}^\intercal\mathbf{I}(\mathbf{z}+\mathbf{y})\leq 0\\ z_i,y_i\in\{0,1\}\end{array} \tag{4.49}$$

For OPT2 we replace the binary slack variables that identify the happiness with the real valued

slack variables y_i that identify the amount the advertisers lose:

$$\begin{array}{l} \min_{\mathbf{z},\mathbf{y}} \bar{1}^{\intercal}\mathbf{y} \\ \text{s.t.} \quad \mathbf{Az} + \mathbf{b} + \mathbf{m}^{\intercal}\mathbf{I}(\bar{1} - \mathbf{z}) + \mathbf{y} \geq 0 \\ \mathbf{Az} + \mathbf{b} - \mathbf{M}^{\intercal}\mathbf{Iz} - \mathbf{y} \leq 0 \\ x_i \in \{0, 1\} \\ y_i \geq 0, \;\; y_i \in R \end{array} \tag{4.50}$$

We derive the ILP versions of OPT3 and OPT4 from OPT1 (4.49) and OPT2 (4.50). The objective function of OPT3 and OPT4 is the total number of clicks. It can be expressed as a linear function of the advertisers' decisions:

$$\min_{\mathbf{z},\mathbf{y}} \mathbf{c}^{\intercal}\mathbf{z} \tag{4.51}$$

We keep all the constraints of OPT1 and OPT2. The former objective function becomes a new constraint:

$$\bar{1}^{\intercal}\mathbf{y} \leq \epsilon \tag{4.52}$$

$$\tag{4.53}$$

Simulated annealing + ILP. In the hybrid approach we use the simulated annealing to get the approximate solution of the problem. Then we use ILP to refine this solution. This allows us to balance the accuracy and the computation time.

The idea is that the simulated annealing can find the values of the majority of the variables correctly. If we know the decision of the advertiser we can remove the corresponding variable z_i from the ILP. If we know the decisions of many advertisers then the ILPs formulated above become manageable in size. Various criteria can be used to select the advertisers whose decisions need refinement. We refine the decisions of those advertisers who are unhappy in the simulated annealing solution.

Optional preprocessing: iterated elimination of dominated strategies. The iterated elimination of the dominated strategies is a useful preprocessing step that reduces the problem dimensionality. It can be used with either of the optimization algorithms. The idea behind the algorithm is that the rational players do not choose the strategy if there is another strategy that is always better (no matter what other players choose). Such dominated strategies are removed from the search space. The process is repeated, because in the reduced space new dominated strategies may appear. The iterativeness has the following interpretation: the player knows that his competitors are rational and do not play dominated strategies; the player computes his best actions based on this knowledge.

The iterated elimination is quite useful in our setup because there are lots of players with dominating strategies. In addition, such players can be discovered very efficiently. The player

has a dominating/dominated strategy if either of these conditions is satisfied:

$$\min_{\mathbf{z}} \Delta\tilde{f}_i(\mathbf{z}) = \min_{\mathbf{z}} \mathbf{a}_i^\intercal \mathbf{z} + b_i = \sum_i \min\{0, a_i\} + b_i > 0 \tag{4.54}$$

$$\max_{\mathbf{z}} \Delta\tilde{f}_i(\mathbf{z}) = \max_{\mathbf{z}} \mathbf{a}_i^\intercal \mathbf{z} + b_i = \sum_i \max\{0, a_i\} + b_i i < 0 \tag{4.55}$$

The first condition suggests that the strategy $z_i = 1$ is dominant and therefore $z_i = 0$ is dominated and can be eliminated from the search. The second condition suggests that the strategy $z_i = 0$ is dominant. The decision of a player becomes a constant. To update the model of utilities, we effectively merge the column $\mathbf{a}_i$ that corresponds to this player with the intercept term:

$$\mathbf{b}' \leftarrow z_i \mathbf{a}_i + \mathbf{b} \tag{4.56}$$

The downside of the approach is that by eliminating the dominated strategies we can eliminate the true optimum. The reason is that in the ϵ-equilibrium (unlike pure and mixed strategy Nash equilibria) the players are allowed to be irrational. We use this preprocessing in our experiments because its error is low compared to the errors introduced by other components of the pipeline: click predictions, utility approximation and the approximate optimization algorithms.

4.4 Results

To evaluate our approach we use the dataset from chapter 2, described in details in section 2.7. Our model consists of two components: regression to approximate the utility functions and Nash equilibrium algorithm for linear utilities. We will evaluate each component independently as well as the end-to-end system.

4.4.1 Model of Utilities

To estimate the local utility approximations we use 5000 data points for training, 5000 for development (cross validation) and 5000 for testing. The models that we train are local, the data is sampled around the solution of the optimization problem (algorithm 1). We sample a separate dataset for each of the problems OPT1-4 and for each value of the Lagrange coefficient for OPT3 and OPT4. The training set size is selected to yield the optimal combination of accuracy and training time. We train 5614 regression functions for each problem (2 models for each of 2807 advertisers). The dimensionality of the feature space is 2806 (the decisions of all competitors). We use LASSO regression [65] trained with LARS package [27] as our learning model. It proved to produce the best results. LASSO has a regularization parameter that determines the model complexity. We tune this parameter on the validation set. Sparseness of LASSO solution

Table 4.1: Utility prediction error

Model	Auctions	GD
Constant model	0.266	0.042
local LASSO	**0.065**	**0.015**

is beneficial to us as it leads to a sparse design matrix of ILP. We use the relative absolute error micro-averaged for all advertisers as our evaluation metric:

$$Error = \frac{\sum_{i,j} |f_i(\mathbf{z}_j) - \tilde{f}_i(\mathbf{z}_j)|}{\sum_{i,j} f_i(\mathbf{z}_j)} \tag{4.57}$$

the summation is over all advertisers and over all test instances. In addition to LASSO, we report the results of a simple constant model, see the table 4.1. The prediction accuracy for GD advertisers is much higher. It is not surprising because GD advertisers target specific click numbers as their campaign goals. Our allocation algorithm can satisfy these goals. The prediction for the auction advertisers is a more challenging task. *This observation supports one of the motivations of this work: the long term performance of keyword auctions is hard to control.* Yet, 6.5% error for the auction advertisers is an impressive result.

4.4.2 Optimization Accuracy

In section 4.3.3 we proposed three variants of optimization for the approximate problems OPT1-OPT4: simulated annealing, exact ILP and the hybrid approach. ILP solves the problem exactly. The other two methods solve it approximately. ILP is actually too slow to handle our problem and we only report the objective of the other two methods. We report the results for OPT1 and OPT2 in the table 4.2. The hybrid approach outperforms the simulated annealing. The results for OPT3 and OPT4 agree with this conclusion. We do not report these numbers in the table to maintain its readability. OPT3 and OPT4 depend on the regularization parameter λ, we would need to report the performances for all values of lambda that we tested.

For neither of OPT1 or OPT2 we know the true objective. Is it worth trying to improve the optimization algorithm? We experimented with the problems of a smaller size that can be solved in reasonable time by ILP exactly. The results of these experiments suggest that there is some space for improvement. However, our further results for the end-to-end evaluation demonstrate that the utility approximation error has stronger effect on the overall performance than the optimization error. That is, *the optimization results are good enough for the current level of approximation*.

Table 4.2: Minimization objective, achieved by different optimization algorithms

	Similated annealing	Hybrid
OPT1	0.0061 ± 0.0006	0.0052 ± 0.0008
OPT2	54.3 ± 12.4	48.7 ± 9.8

Table 4.3: End-to-end evaluation

Metric	Random	RL baseline	OPT1	OPT2
Max loss (in clicks)	10,500	16,300	257	**221**
% unhappy advertisers	32.6%	21.5%	**5.5%**	8.3%
Total losses (in clicks)	53,300	34,400	**2080**	2227
Revenue ($\times 10^7$)	1.77	1.76	**1.82**	1.79
Clicks ($\times 10^5$)	4.04	3.99	**4.19**	4.18
Cost per Click	43.8	44.1	43.4	**42.8**
Delivery rate	**0.866**	0.842	0.835	0.832

4.4.3 End-to-end Equilibrium Evaluation

Finally we compare the performance of the whole framework. We estimate its ability to find good approximations of Nash equilibrium. For each method we report the percentage of unhappy advertisers (directly optimized in OPT1), the sum of losses of these unhappy advertisers (directly optimized in OPT2) and the maximum loss of a single advertiser. The evaluation metrics are computed exactly.

We compare different variants of our framework with the reinforcement learning (RL) baseline [50] and the advertisers behaving randomly. The results of this evaluation for OPT1 and OPT2 are presented in Table 4.3. As we can see our method significantly outperforms the baseline in all equilibrium related metrics. It is surprising, that the method OPT1 performs better than OPT2 even though it has weaker asymptotic guarantees.

The metrics that characterize the social welfare of the equilibria are reported in the rows 4-6 of Table 4.3. We use the same metrics that we used in the previous chapters: the revenue, the number of clicks, the average cost per click and the delivery rate for GD advertisers. As we can see both OPT1 and OPT2 find the states with very good social welfare. Neither random guessing nor reinforcement learning allows the advertisers to converge to a good quality strategy. The

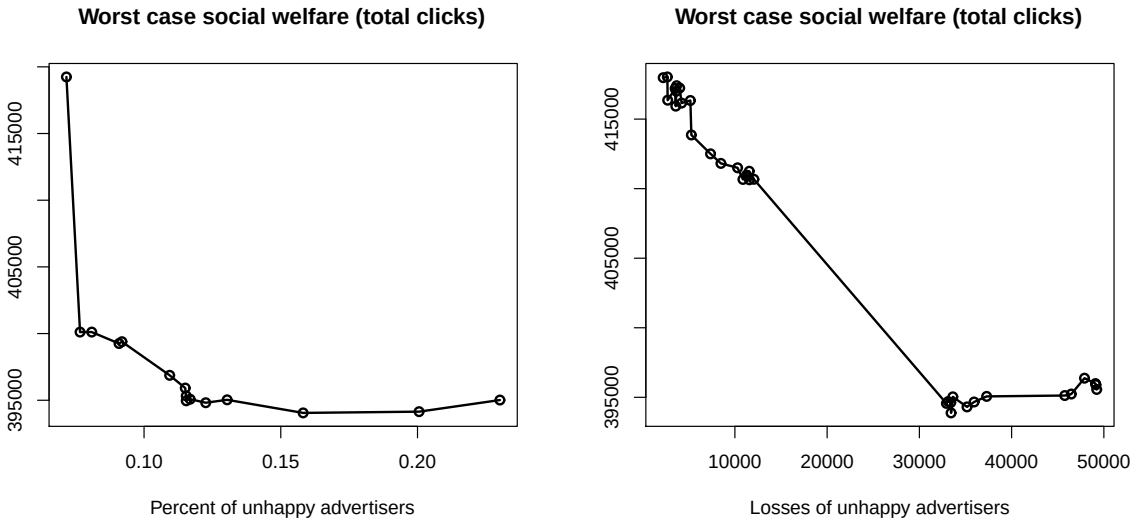


Figure 4.2: Worst case social welfare (number of clicks) discovered by OPT3 (Left) and OPT4 (Right). The x-axis represents allowed deviation from the equilibrium. For OPT3 the deviation is measured as the percentage of unhappy advertisers, for OPT4 it is measures as a sum of losses of unhappy advertisers. The left-most points correspond to the solutions of OPT1 and OPT2 solutions respectively.

strategies computed by our algorithm have clear advantage. Only the search engine can discover these strategies, the advertisers cannot do it due to the lack of information. This emphasizes the important role of the search engine: *the global optimum can be only achieved through the collaboration of the advertisers and the search engine.*

The methods OPT3 and OPT4 find the worst case social welfare (number of clicks) for a given deviation from the equilibrium, ϵ. This metrics show us what can happen with the performance if the advertisers start to explore the strategies around the equilibrium proposed by OPT1 and OPT2. The left plot on the figure 4.2 presents the results of OPT3: the worst case social welfare given the number of unhappy advertisers. The leftmost point on this curve corresponds to the OPT1 equilibrium. As we deviate from the OPT1 solution the worst number of clicks drops very fast. This result has a natural interpretation: it is enough for a few large and influential advertisers to choose suboptimal strategies to spoil the overall performance dramatically. The right plot on the figure 4.2 presents the performance of OPT4: the worst case social welfare given the sum of losses of unhappy advertisers. We do not see any sudden drops of performance as we go further from the equilibrium. If the player chooses a suboptimal strategy, but not much worse than his optimum, than his decision won't have strong effect on the overall performance.

Stability is a good property of the algorithm. The lack of the measurements in the middle region of the right plot is the result of the Lagrange method. This middle region corresponds to a very narrow interval of the Lagrange multiplier. In other words the dual function has a singular point.

Finally, on figure 4.3 we summarize the social welfare performances of all the results above. We align these results with the evaluation charts from the chapter 2. OPT1 and OPT2 are special cases of OPT3 and OPT4. We can easily identify the points that correspond to OPT1 and OPT2: they yield the highest revenue and the highest number of clicks.

To summarize our experimental findings, the equilibrium approximation framework proposed in this chapter successfully fulfills its task. The game theoretic analysis provides more insights into the performance of the joint auction and GD allocation algorithm. We can refine the evaluation results obtained in the previous chapters. The new results strongly support our approach.

4.5 Summary

In this chapter we present a framework to find approximate Nash equilibrium for the strategic decision between auctions and guaranteed delivery in sponsored search. This analysis allows us to perform accurate evaluation of our allocation algorithm and to provide optimal strategy recommendations to the advertisers.

We present two variants of equilibrium approximation that minimize the number of unhappy advertisers (OPT1) and minimize the sum of losses of unhappy advertisers (OPT2). Further, we introduce the formulations OPT3 and OPT4 to analyze the worst case social welfare for a given deviation from the equilibria. The advertisers' utility functions in our formulations cannot be expressed in closed form. To address this complication we use local linear approximations of these utilities. We establish the theoretical approximation bounds for the problems OPT2 and OPT4.

The setup used in this work (several thousands of players and unknown utility functions) has not been investigated before, therefore no representative baselines exist. We compare our equilibria with the equilibrium obtained by the reinforcement learning. The experiments performed on the data obtained from Bing search engine demonstrate that our algorithm finds significantly better approximations of Nash equilibrium.

The discovered strategic profiles have high social welfare: both the total number of clicks and the revenue of the search engine. This means that there exist states that are jointly beneficial for the search engine and the advertisers. Of course, the advertisers cannot arrive to such states due to the lack of information, as demonstrated by the reinforcement learning baseline. The search engine, on the other hand, possesses information required to compute such equilibrium. Our future goal is to design a communication protocol between the search engine and the advertiser (for example, through feedback reports and recommendations) that allows the system to arrive to

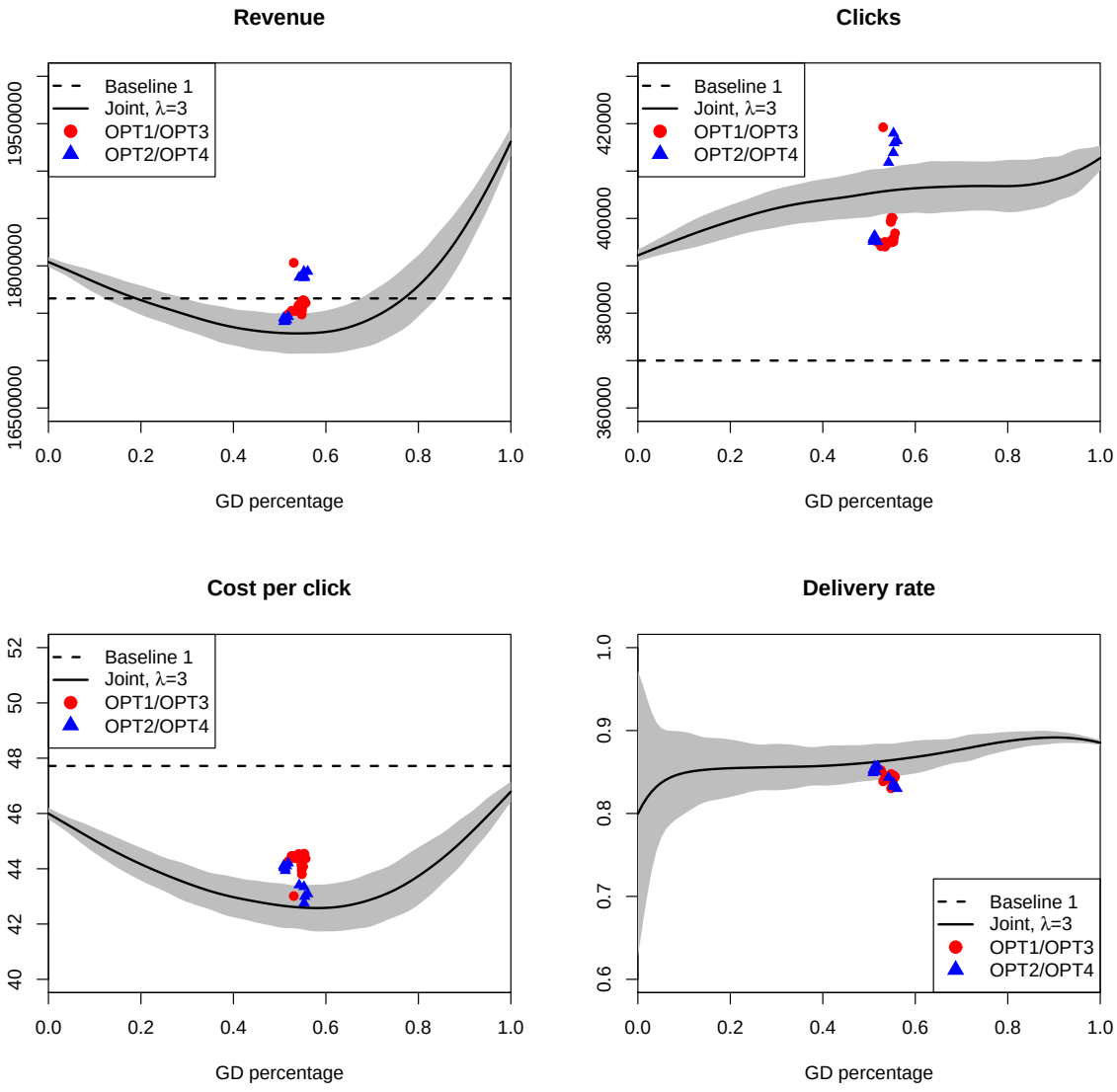


Figure 4.3: OPT1-4 equilibria performance measured for the selected metrics: the revenue, the number of clicks, the cost per click and the delivery rate. The solid curve corresponds to chapter 2 evaluation for random advertisers. The red point that stands out from the cluster corresponds to OPT1.

the desired equilibrium.

The framework proposed in this chapter is not specific to the ad allocation algorithm. It can be used to analyze any situation when the advertisers have to make a binary strategic choice, for instance, selection between two advertising platforms. In future we plan to extend the algorithm to multi-choice situations and to mixed strategy Nash equilibrium.

Chapter 5

Wind farm layout optimization

5.1 Introduction

In this chapter we investigate the problem of wind farm layout optimization. What links this problem to the previous chapters is the demonstration of the generality of the hierarchical decomposition technique.

Due to the significant increase of the energy consumption the renewable energy became a hot topic in the last decades. For instance, the worldwide generation of wind energy is growing at 20-25% annually [8]. As of 2010 the wind energy accounted for 2.5% of the overall energy production [5]. Increasing the economic efficiency of wind energy production is one of the major goals of the industry. We apply the hierarchical optimization technique to find the optimal placement of the wind turbines and to generate the maximum power output. We address several serious limitations of the existing techniques: the exponential complexity and the inferior model of the power output.

A typical wind farm consists of multiple wind turbines with some infrastructure (roads, cables, etc.) The number of turbines in a farm can vary from one to hundreds. To limit the costs of the land and the infrastructure the turbines should be packed compactly in the regions with strong and consistent wind. The availability of such regions is also limited. The turbines cannot be placed too close to each other because of their physical dimensions (around 100 meters in diameter for commercial 2-3 MWatt turbines) and because of the wake effect, i.e. the reduction of the wind speed caused by the turbines. Finding the best locations for the turbines poses a complex constrained optimization problem. What makes this problem so challenging is the interference of the turbines due to the wake. First of all, encoding such interference into an optimization problem leads to combinatorially complex solutions. Second, the accurate power output models that account for the wake losses are usually too complex to be used in the optimization directly and therefore less accurate *linear superposition models* are used in the existing

optimization approaches. In this work we address these two unsolved problems and introduce a new hierarchical decomposition algorithm. We segment the wind farm location into multiple regions and formulate the master optimization problem that decides how many turbines to install in each of these regions. The allocation of the turbines inside each region is done independently of other regions by a child optimization program. The master problem takes into account the power output of each region and the interference between regions. Such decomposition provides great potential for parallelization and significant reduction of the problem dimensionality and hence addresses the first issue, the scalability. To address the second problem, namely the complexity of the power output models, our solution incorporates the accurate power model to compute the output of each child problem and to compute the interference between every pair of regions. We still use the linear superposition model compute the aggregate effect of several regions. We test our solution for wind farms located on flat terrains.

Little work has been done on optimization frameworks for the wind farm layouts. However, the importance of the effective and efficient optimization solutions is acknowledged by other researchers [28, 29, 60]. Currently most decisions in the industry are made in an adhoc manner using very simple rules that lead to regular rectangular layouts. Such adhoc decisions can lead to bad decisions even for simple one turbine layouts. For example, a township had to give up a wind turbine due to insufficient wind [57]. Needless to say, the layouts for multi-turbine farms chosen is this manner are very likely to be suboptimal. The existing principled solutions mostly rely on the heuristic optimization algorithms: genetic algorithms and simulated annealing (e.g. [49])[1]. More recent approaches are based on integer linear programming (ILP) and mixed integer linear programming (MILP) [24, 29]. The idea is to select the locations of the turbines from a finite set of the nodes of a coordinate grid, presence or absence of a turbine in the node is modeled by a binary variable. However, these methods do not scale. The major problem is the wake effect: each turbine extracts energy from the wind and therefore reduces the wind speed behind it. Modeling the wake in the ILP greatly increases the number of variables and the constraints: the number of "wake" variables is quadratic with the number of the nodes of the grid. In [29] the authors formulate the wake component of the model, but mainly test their approach without it. In [24] the authors restrict the wake effect to the direct proximity of the turbine. The neglect of wake or the short-distance wake simplifies the problem, but loses much of the purpose of the optimization. The other limitation of the ILP solutions is the unavoidable tradeoff in the selection of the grid density. High density grids allow more accurate placement of the turbines but increase the number of the variables making the problem not tractable. Finally, the existing solutions rely on a non-accurate linear superposition wake model (in order to apply linear programming). In our experiments we have discovered that this model is reasonably accurate for small farms with

[1]This is likely to be true for commercial solutions as well, though their algorithmic details are not always available. For instance, [29] describes a greedy heuristic approach used in a popular package WindPro 2.6 (The current version is 2.9 http://www.emd.dk/WindPRO/Frontpage).

Figure 5.1: Left: horizontal axis offshore wind turbine. Right: vertical axis wind turbine. Images source: http://en.wikipedia.org/wiki/Wind_turbine.

few turbines, but fails for large farms with densely installed turbines.

In this work we address the limitations of the existing ILP solutions mentioned above. Our hierarchical solution gives us the opportunity to model the other important factors not considered in the previous solutions. One of such factors is the periodical (daily, seasonal) variation of wind direction. Most large modern wind turbines can adjust to wind direction (yawing) and the wind farms optimized for the single prevalent wind direction become sub-optimal. Another factor is mixing the turbines of different types in a single farm. For instance, installation of the turbines of different heights can reduce interference. This extension can be handled by our solution, though we use the turbines of one type in our experiments.

The chapter is organized in the following way. In section 5.2 we provide the necessary background on the wind farm modeling and the related work. In section 5.3 we describe our optimization algorithm. We report our experimental results in section 5.4 and summarize the work in section 5.5.

5.2 The Essentials of Wind Energy

In this section we give the necessary background on the wind energy. We describe the properties of wind, the characteristics of a typical commercial wind turbine, and the popular models of wake that we use in our solution. We also outline the existing ILP solution that becomes a component of our framework.

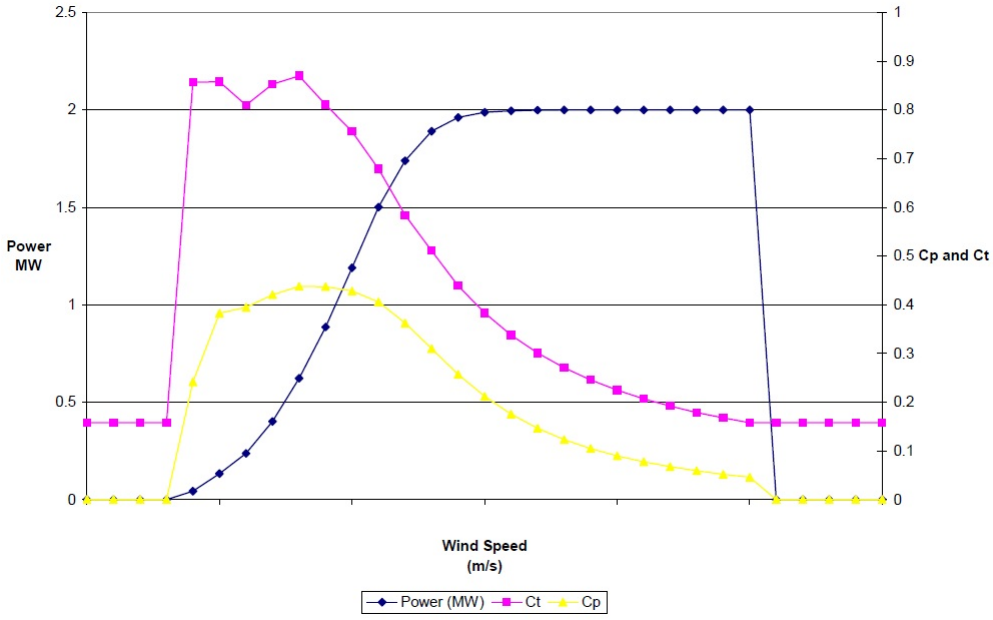


Figure 5.2: Power curve, power coefficient (C_p) and thrust coefficient (C_t) of a 2MW wind turbine (Bonus Energy A/S) [2]

5.2.1 A Wind Turbine

A wind turbine is the main building block of a wind farm. A turbine can have horizontal axis construction (left of figure 5.1) or a vertical axis construction (right of figure 5.1). The vertical axis turbines do not require alignment with the wind but due to technological limitations are not used in large wind farms. The horizontal axis turbines, used in the most large wind farms, must be pointed into the wind for the maximum power output. Modern turbines usually have a motor to rotate them around the vertical axis.

The main parameters of a wind turbine are its cost, physical dimensions and the power output function. The cost includes the costs of the turbine itself, its installation and the maintenance. The budget is always one of the optimization constraints. The physical dimensions of a turbine determine the placement constraints and the geometry of the wake. The diameter of the rotating part (the rotor) usually ranges from 70 meters to 150 meters. The minimum distance between turbines is usually taken as 3-5 rotor diameters.

The power output of a turbine is a function of the wind speed. The output of the turbines used in most large wind farms ranges from 1.5 MW (Megawatt) to 3 MW. The curve is usually characterized by the cut-in wind speed - when the turbine starts generating energy; the cut-out wind speed - when the turbine is shut down due to the safety reasons; the nominal wind speed -

Table 5.1: Specifications for a 2MW Bonus Energy A/S turbine.

Hub Height	60m
Rotor Diameter	76m
Area Swept	$4526m^2$
Cut-in Speed	4m/s
Cut-out Speed	25m/s
Standing Thrust Coefficient	0.158

the speed at which the turbine starts generating the nominal power output. When the wind speed exceeds the nominal wind speed the controller adjusts the pitch of the blades to keep the power output equal to the nominal output. A typical power generation curve is displayed in figure 5.2. Another characteristic of the turbine, displayed in figure 5.2, is the thrust coefficient (C_t). We do not give the detailed explanation, in simple words, this coefficient measures the proportion of the energy extracted from the stream and is used in the model of the wake discussed further in the section. In the experiments we use the parameters of a representative 2MW turbine from [2], table 5.1.

5.2.2 Wind Speed and Direction

The power output of the wind farm depends on wind speed and direction. To optimize the wind farm layout it is essential to know the distribution of wind speed and direction over time. Prior to the installation of the farm the territory is monitored for the wind conditions for at least one year. This gives sufficient information about daily and seasonal variations of wind speed and direction. The distribution of the wind speed over time can be approximated by the Weibull distribution [70] (figure 5.3) with the following density function:

$$f(v|\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{v}{\lambda}\right)^{k-1} e^{(v/\lambda)^k} & v \geq 0 \\ 0 & v < 0 \end{cases} \tag{5.1}$$

Varying wind direction can affect the optimal layout of a farm. It is not uncommon for the wind to blow uniformly within a wide interval of directions. For instance, in the location investigated in [28] the wind is equally strong throughout a year in a 90-degree sector (from North-West to South-West). Many existing solutions assume one prevalent wind direction. These solutions became too limited with the spread of the wind turbines with yawing ability.

Finally, wind speed may vary from one location to another location of the same farm, especially within the onshore wind farms located on rough terrains, e.g. mountain ridges. Landscape,

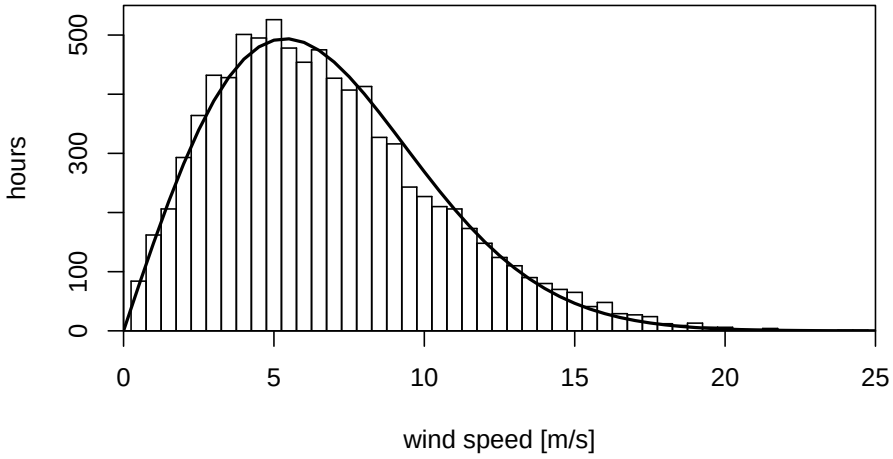


Figure 5.3: Distribution of wind speed for all of 2002 at the Lee Ranch facility in Colorado [31]. The histogram shows measured data, while the curve is the Weibull model distribution for the same average wind speed and $k = 2$.

buildings and even trees can affect the wind speed. For flat surfaces and offshore farms the wind speed is usually considered to be uniform over the location.

5.2.3 Jensen's wake model

A wind turbine extracts energy from an air flow, which results in reduction of the wind speed in the downstream. The wind deficiency due to the turbine interference is called *wake*. The wake effect can result in 10-20% losses of power production. This is a significant amount and it is important to incorporate the model of wake into the layout optimization. Unfortunately, installation of each turbine affects the wind speed around it and consequently affects the power output of other turbines in the area. This leads to complex interdependencies between turbines. These interdependencies are the major reason for complexity of the layout optimization problem.

A number of wake models of various complexities exist: from the solutions based on computational fluid dynamics to simple heuristic formulas. In this work we use Jensen's wake model [36]. It is a simple model that fits the measurement data very well, in fact, according to [63], this model outperforms several more complex alternatives. Its simplicity allows us to use it in the optimization. The Jensen's wake model assumes that the wake region spreads linearly

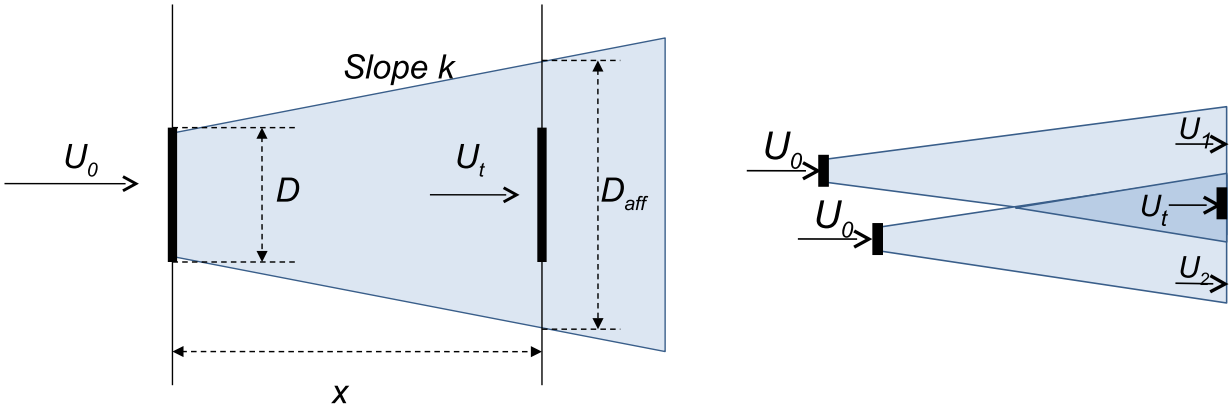


Figure 5.4: **Left:** Jensen's wake model. U_0 - unaffected wind speed, U_t - the wind speed at downstream location, D - the diameter of a rotor, x - distance from the turbine, affected cross section area extends linearly: $D_{aff} = D + 2kx$. **Right:** Superposition of several wakes.

behind the turbine and the wind deficiency reduces with the distance from the turbine at a square rate, left of figure 5.4. According to the Jensens model, the wind speed in the wake is reduced by:

$$\Delta U = U_0 - U_t = U_0 \frac{1 - \sqrt{1 - C_t}}{\left(1 + \frac{2kX}{D}\right)^2} \tag{5.2}$$

The parameters of the model are summarized in table 5.2. The computation of the superposition of multiple wakes (right of figure 5.4) can be done in multiple ways. Linear superposition models and quadratic superposition models are often used:

$$\Delta U = \sum_{i=1}^{N} \Delta U_i \tag{5.3}$$

$$\Delta U^2 = \sum_{i=1}^{N} \Delta U_i^2 \tag{5.4}$$

where ΔU_i is the deficit of each wake, computed with single wake Jensen's model (5.2). Linear superposition is easier to model, this is especially important for linear programming based solutions (they are all based on this assumption). However, the sum of squared deficits (5.4), proposed in [40], is much more accurate. We will use the quadratic model to compute the reference power output of the layout. We also use it for several components of our optimization algorithm and this is one of the contributions of this work. Compared to the squared model, the

Table 5.2: Parameters of Jensen's wake model

U_0	unaffected wind speed
U_t	downstream wind speed
C_t	thrust coefficient of the turbine
k	wake decay coefficient
X	distance downstream
D	diameter of upstream turbine

linear model, used in the baseline optimization approaches, overestimates the wake loss. This overestimation leads to some negative unexpected effects that we discuss in the results section.

Wake decay coefficient, k, has a strong influence on the outcome of the model. This coefficient characterizes how intensively the wind streams mix with each other in the wind farm location. The coefficient depends on multiple factors: surface temperature, air temperature, roughness of the surface, turbulence etc. Larger value of k means faster decay, i.e. the wake mixes with the ambient wind faster. Most of the times the value lies between 0.03 and 0.08 and it is almost impossible to predict it with higher accuracy. The values of open land wind farms are usually higher and the values for offshore farms are lower. For instance, the default values in the commercial software WindPRO are respectively 0.04 and 0.75 [63]. However, the study of several wind farms in [63] shows that this assumption may be wrong. The problem of choosing the right value of k is orthogonal to the problem that we solve in this chapter, unless stated otherwise we use $k = 0.04$ more typical for the offshore farms.

The wake effect can result in 10-20% losses of power production in a typical wind farm. To give a better quantitative understanding of power losses in the wake and the range of the wake, we perform a simple case study. We investigate the wake produced by a row of 33 turbines. The distance between turbines is 300 meters and the whole row is 10 kilometers in length. The wind blows at a constant speed of 10 m/s in the direction orthogonal to the row. To get a smooth realistic image we allow the wind direction to vary from -10 to 10 degrees from the main direction. We use the turbines mentioned above (table 5.1), the decay coefficient k is set to 0.04. In figure 5.5 we show the map of the wind speed loss. The wake is quite noticeable even several kilometers away from the row. In the right part of the same figure we measure the power loss (solid line) of a turbine installed in the affected area at a certain distance from the row. At 5 kilometers away from the row the loss in power production is 2.5%. We compare the loss inflicted by a row of turbines with the loss, inflicted by a single turbine (dashed line). As we can see, the two curves differentiate at 1km (when the wakes begin to overlap). At 5km the power loss inflicted by a single turbine is approximately 0.6%. This example demonstrates the importance of modeling the long-distance global interference between the turbines as opposed to local interference adopted

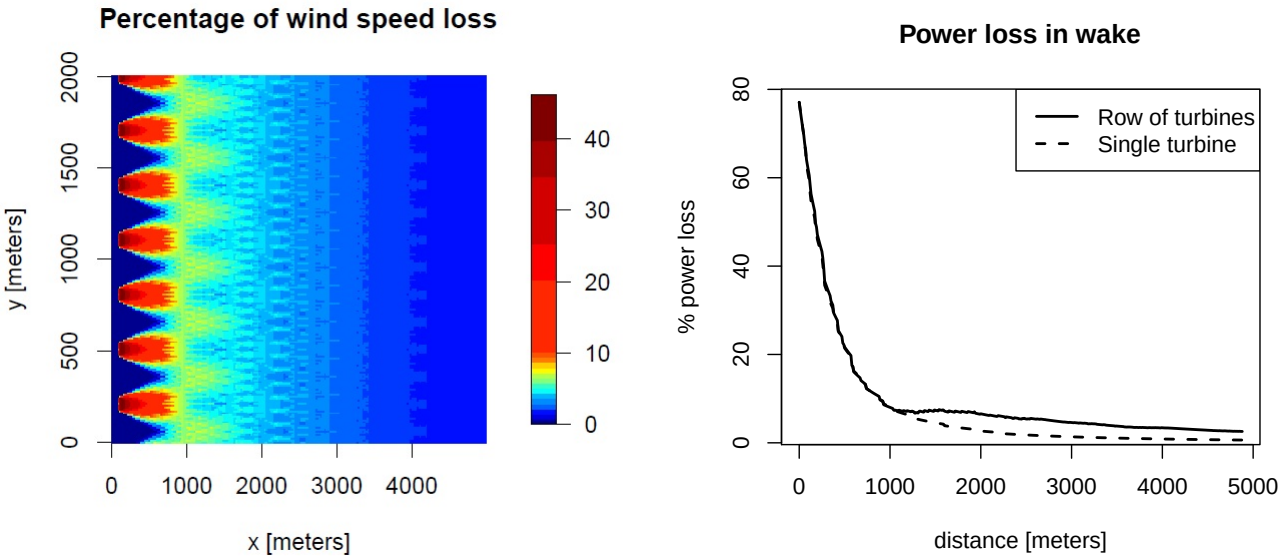


Figure 5.5: **Left:** a wind map for the wake, generated by a row of 33 turbines, the distance between the turbines is 300 meters. **Right:** The power loss of turbine installed in the wake of the row of turbines (solid line) or in the wake of a single turbine (dashed line). The wind speed is 10 m/s, the direction ranges from -10 to 10 degrees.

by many existing solutions.

5.2.4 Integer Linear Programming for Layout Optimization

Recently, integer linear programming became a popular optimization technique for wind farm layout optimization [24, 29, 60]. Compared to simulated annealing and genetic algorithms often used for wind farm optimization, ILP has stronger theoretical foundation. ILP is attractive because it provides quite rich modeling abilities and multiple out of the box solvers. On the other hand, ILPs with binary variables belong to the class of NP-complete problems. This means that they are very likely not solvable in polynomial time. Whether the practical problem can be solved in reasonable time depends on its size and structure. Our analysis of realistic wind farm setups suggests that the layout cannot be efficiently optimized with a generic ILP.

We outline the main ideas of the ILP approach and demonstrate its limitations now. A detailed derivation of this model as a part of our solution is given in the next section. ILP finds the optimal location of the turbines on a discrete coordinate grid. Each binary variable x_i corresponds to a node in the grid and tells whether this node has a turbine. The objective of the optimization is

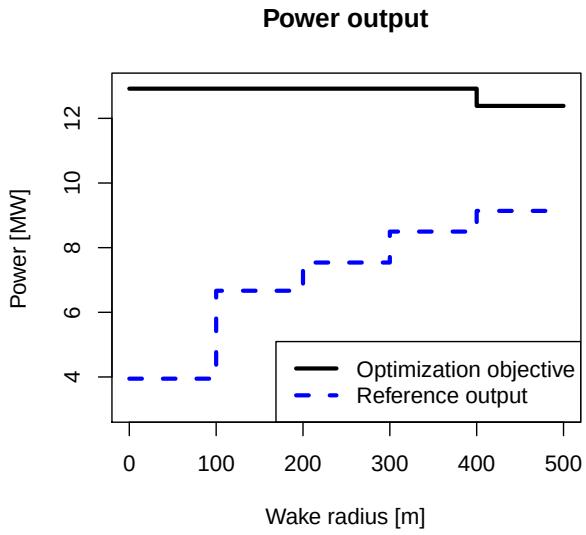
the power output of the farm. The wake affects the power output of the turbines. If the turbine in the node i affects the turbine in the node j, then then power output of the turbine j is reduced by a constant value a_{ij} (computed by Jenson's formula 5.2). If several turbines affect the turbine in the node j, then the effect on the power output is assumed to be additive (the assumption about the linear superposition of wakes in action): $a_{i_1 j} + a_{i_2 j} + ...$ The linear program is formulated in the following way:

$$\begin{aligned} \max_{\mathbf{x},\mathbf{y}} & \sum_{i \in Nodes} c x_i - \sum_{\{i,j\} \in Edges} a_{ij} y_{ij} \\ \text{s.t.} \quad & y_{ij} \geq x_i + x_j - 1 \quad \forall \{i,j\} \in Edges \\ & \sum_{i \in Nodes} x_i \leq N \\ & x_i, y_{ij} \in \{0,1\} \end{aligned} \tag{5.5}$$

In the formulation above the objective is the sum of unaffected turbine outputs (c is the output of an unaffected turbine) minus the interference penalties. The edges $\{i,j\}$ denote potential turbine interference, i.e. the node j is located in the wake of the node i. The constraints guarantee that $y_{ij} = 1$ when both nodes have the turbines, i.e. the interference actually happens, and $y_{ij} = 0$ otherwise. The last constraint limits the number of the turbines. Some minor modifications are common in this formulation, for instance, the proximity constraints, i.e. two turbines cannot be installed in the adjacent nodes. The following is the list of limitations of this model:

- The accuracy of the allocation is determined by the resolution of the grid. Higher resolution leads to more nodes and larger dimensionality of the problem.
- The number of the edges corresponds to the number of interfering nodes and is even more critical for the problem size and tractability. Usually, to make the problem solvable only short range interference is considered, not further than several diameters of the turbine. As we saw in the case study above this assumption is far from being realistic.
- The penalty coefficients a_{ij} depend on the ambient wind speed through the thrust coefficient C_t which depends on the wind speed, figure 5.2. The existing works assume that the wind speed is constant.
- Constant wind direction is also assumed.
- The penalty caused by the overlapping wakes is additive. As we mentioned earlier, the additive model is not accurate.

The first four limitations are directly related to the scalability of the solution. We address those issues with our hierarchical optimization approach. In order to demonstrate that the scalability is actually a problem, we perform another case study. We specifically want to emphasize the effect that the wake model has on the accuracy and the computation time. The setup is the



Wake radius [m]	Timing [sec]
0-399	0
400-499	1,571
500-599	>22,000

Figure 5.6: The objective of the ILP optimization for different wake distances. The reference power output is computed with the exact Jensen's model. On the right are the timings for different settings. The optimization for $r > 500$ did not complete and was forcefully terminated after 22,000 seconds.

following, we use a 9 by 9 coordinate grid, the resolution is 100 meters, i.e. the whole region is 800 by 800 meters. The resolution is comparable with the diameter of the rotor (76 meters). This is a reasonable choice, i.e. the accuracy of the placement is approximately equal to the size of the turbine. The wind speed is constant (10m/s), but has two possible directions: south 50% of the time and west 50% of the time. To investigate the influence of the wake model in the formulation above, specifically the maximum wake radius, we compare the optimization objective with the actual power output computed with the exact Jensen's wake model (reference output). Ideally, the objective and the true output should be the same. In reality the objective of the ILP is not accurate because of the simplifications listed above. As we increase the maximum wake radius, the optimization becomes more accurate and the reference power output increases, see the figure 5.6. Up to a certain point (400 meters) there exists a non-penalized ILP solution that the optimizer finds in a fraction of a second (see the timing statistics on the right of the figure 5.6). When the interference radius is more than 400 meters, the non-penalized solution does not exist anymore. The running time increases immediately from 0 to 1,571 seconds. After 500 meters the computation time increases even further. In fact, after 22,000 seconds the solver was still running and we terminated it forcefully. Figure 5.7 demonstrates how the turbine allocation looks for different

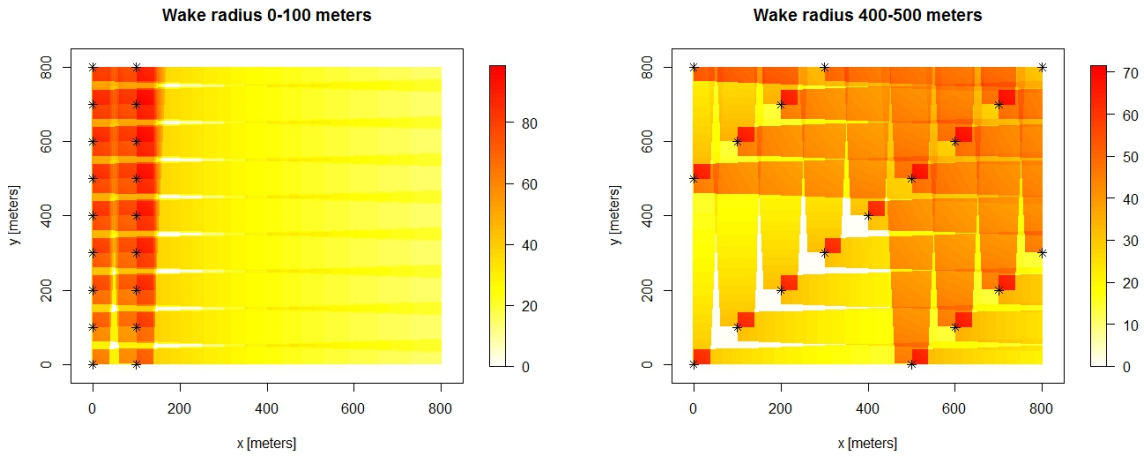


Figure 5.7: The allocation of the turbines for different maximum wake radius in the case study: 0-100m on the left and 400-500m on the right. The color map measures the wind speed loss in percent.

wake radiuses. We learn several important lessons from this case study. First, modeling long distance interference is essential to achieve high power output. Second, the computation time for longer interference increases exponentially fast. This motivates our novel solution, presented in the next section.

5.3 Optimization Algorithm

The algorithm presented in this section is based on the idea that short-distance interactions and long distance interactions can be modeled differently. The long distance interactions describe the interference between the clusters of turbines, while short-distance interactions describe the interference between turbines within the cluster. We also formulate these two types of interferences using hierarchical optimization framework, in a way, similar to the ad allocation problem. The master problem decides how many turbines to install in a specific region; the children problems decide how to allocate the turbines within this region. We will first describe the model for short distance interactions which is an extension of the integer linear programming approach, presented in the previous section. Then we describe the model for long distance interactions and our hierarchical optimization approach.

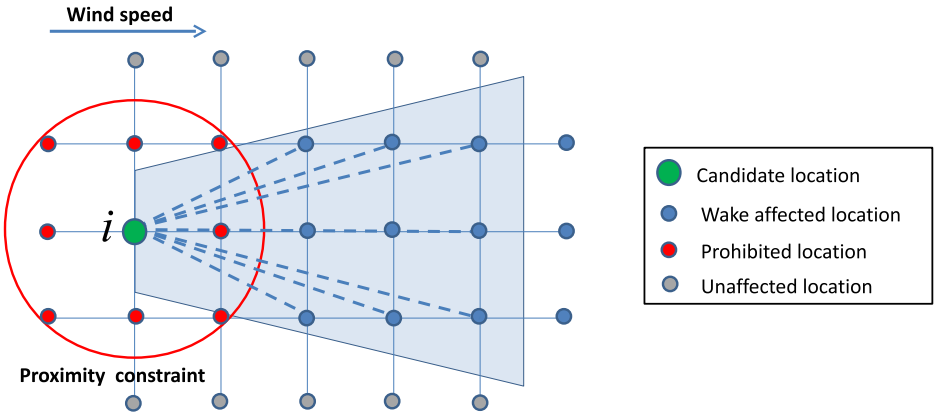


Figure 5.8: Coordinate grid. Each node of the grid is a candidate location for the turbine. Installation of the turbine in the candidate location i affects other nodes. The nearby nodes (red) cannot have turbines, the blue nodes are affected by the wake, and the grey nodes are unaffected.

5.3.1 The Model for Short Distance Interactions

The formulation for the short distance interactions is based on ILP the model, outlined in the previous section. We start with a simple case: the wind speed and the wind direction are constant: u. The locations for the turbines are selected in the nodes of a grid, see figure 5.8. We deal with a limited number of candidate turbine positions: the nodes $i = 1..N$. A binary variable $x_i = 1$ if there is a turbine in the node i and $x_i = 0$ otherwise. We define a set of the wake interference edges W in the following way: if the node j is located in the wake generated by the turbine installed in the node i, then the edge $e_{ij} \in W$. The goal is to represent the power output of the farm as a function of variables x_i. Jenson's wake model (5.2),(5.3) allows us to compute the wind speed u_i for each node:

$$u_i = u\left(1 - \sum_{j:e_{ji}\in W} \frac{1-\sqrt{1-C_t}}{(1+2kl_{ji}/D)^2} x_j\right) \tag{5.6}$$

In the formulation above the wind deficiency depends on all upstream nodes j that can potentially affect the node i (i.e. $e_{ji} \in W$) and that have the turbine installed (i.e. $x_j = 1$). The thrust coefficient C_t depends on the ambient wind speed u, as it can be seen from the figure 5.2. The value of l_{ji} is determined by the geometry of the grid and the wind direction.

The power output of a turbine for a given speed u_i is computed using a reference table provided by the manufacturer, figure 5.2. For small variations of the wind speed, caused by the

wake, the power output function can be considered linear:

$$power(u_i) = p - \alpha \sum_{j:e_{ji} \in W} \frac{1 - \sqrt{1 - C_t}}{(1 + 2kl_{ji}/D)^2} x_j \tag{5.7}$$

Here p is the power output of the unaffected turbine for the ambient wind speed u and α is the slope coefficient. To simplify the expression we introduce the following notation:

$$a_{ji} = \alpha \frac{1 - \sqrt{1 - C_t}}{(1 + 2kl_{ji}/D)^2} \tag{5.8}$$

$$power(u_i) = p - \sum_{j:e_{ji} \in W} a_{ji} x_j \tag{5.9}$$

We maximize the power production of the whole farm that is the sum of the power production of the individual turbines:

$$\begin{aligned} &\max_{\mathbf{x}} \sum_i p x_i - \sum_{e_{ji} \in W} a_{ji} x_i x_j \\ &\text{s.t.} \quad \sum_i c_i x_i \leq C \\ &\qquad x_i \in \{0, 1\} \end{aligned} \tag{5.10}$$

The constraint represents the budget limit, c_i is a cost of the turbine installation for the node i. We get rid of a non-linear term $x_i x_j$ by replacing it with a new binary variable y_{ij} which is equal to one only if both x_i and x_j are equal to one:

$$y_{ij} \geq x_i + x_j - 1 \quad \forall e_{ij} \in W \tag{5.11}$$

Usually, the hard proximity constraints are added to limit the minimal distance between the turbines:

$$x_i + x_j \leq 1 \quad \text{if} \quad Distance(i, j) \leq R \tag{5.12}$$

Essentially, this derives the existing ILP formulation (5.5).

Variable wind speed

The formulation above can be extended for the variable wind speed if we know the distribution of the wind speeds over time: $P(u)$. The objective function of the formulation (5.10) has the coefficients p and a_{ji} that depend on the wind speed and the variables x_i that do not depend

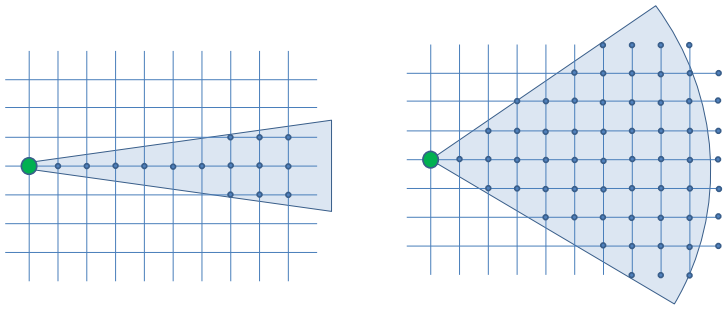


Figure 5.9: Wake of a single turbine for a constant wind direction (left) and non-constant wind direction (right).

on the speed. To extend the formulation it is sufficient to take the expectation of the objective function with respect to the distribution of wind speeds:

$$E_u\left[\sum_i p(u)x_i - \sum_{e_{ji}\in W} a_{ji}(u)x_i x_j\right] = \sum_i E_u[p]x_i - \sum_{e_{ji}\in W} E_u[a_{ji}]x_i x_j \tag{5.13}$$

The expectations $E_u[p]$ and $E_u[a_{ji}]$ for a given distribution are computed using the numeric integration. This procedure affects the values of the coefficients and does not affect the structure of the optimization problem.

Variable wind direction

The problem can be easily adjusted for the variable wind direction just like for the variable wind speed. Conceptually, we take the expectation of the objective function over possible wind directions. However, considering multiple wind directions greatly increases the number of the interference edges $e_{ij} \in W$, see figure 5.9. Solving such a problem using the traditional approach becomes prohibitively expensive.

Mixing the turbines of differen types

Mixing the turbines of different types in one setup is an interesting approach that is often overlooked. The ILP formulation above is general enough to model this opportunity. Again, the computational efficiency is the major bottleneck. The number of variables and the constraints increases linearly and quadratically with the number of turbine types.

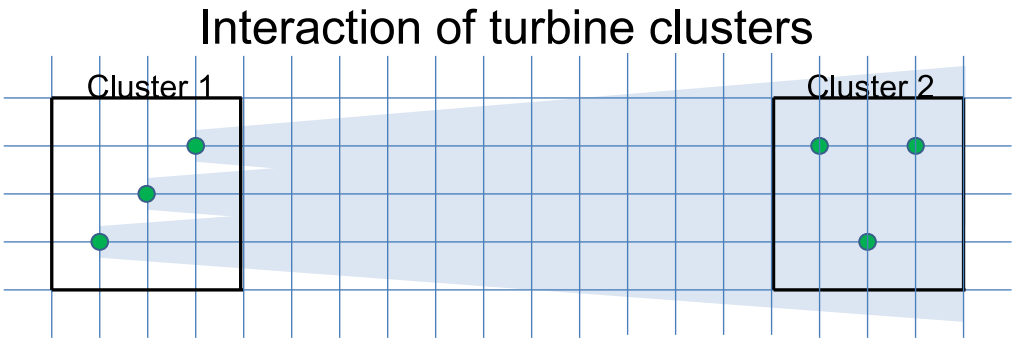


Figure 5.10: The interaction of the clusters of the turbines.

5.3.2 The Model for Long Distance Interactions

The separation of long-range wake interactions from short-range wake interactions is the key idea that allows the formulation of hierarchical optimization. Essentially, the penalty a_{ij} for a pair of interacting turbines does not change much for minor placement variations, as long as these variations are small compared to the distance between the turbines. A similar idea was formulated as a potential future direction in [29] but for a different goal. In [29] the author proposed to make small adjustments for the placement of each turbine, assuming that the wake is approximately constant within this micro-management region. We use this idea to macro-manage the clusters of turbines. We model the interactions of two regions as the interaction of the clusters of the turbines, rather than the interaction of individual turbines, figure 5.10. Essentially, we replace multiple individual wakes with a single wake inflicted by the cluster. This wake depends only on the number of the turbines, installed in the cluster. The variable wind speed actually helps since it acts as a natural smoothing factor and justifies the averaging.

Let us express the power output deficiency caused by the interaction of two clusters C_1 and C_2:

$$P(C_2) = P_0(C_2) - \alpha \sum_{j \in C_2} x_j \sqrt{\sum_{\substack{i \in C_1 \\ e_{ij} \in W}} \left(\frac{1 - \sqrt{1 - C_t}}{(1 + 2kl_{ji}/D)^2} \right)^2 x_i} \tag{5.14}$$

The power output of the cluster C_2 is denoted by $P(C_2)$. The unaffected output of the cluster, $P_0(C_2)$, is computed using the model for close-range interactions. Note, that for long distance interaction we use a quadratic superposition model (5.4) which is more accurate than the linear model. We use quadratic model to model the wakes produced by the turbines of one cluster and liner model to combine the wakes produced by multiple clusters. The unaffected power production of a cluster $P_o(C)$ only depends on the cluster internal structure and is not coupled

with the second term. This is very convenient because this function can be computed once and then reused in the optimization.

According to our assumption the relocation of a turbine within cluster is not significant, therefore we replace the distance between the turbines in different clusters l_{ij} with the average cross-cluster distance l. This simplifies the power output function:

$$P(C_2) = P_0(C_2) - a \sum_{j \in C_2} x_j \sqrt{\sum_{i \in C_1} x_i} \tag{5.15}$$

$$a = \alpha\beta \frac{1 - \sqrt{1 - C_t}}{(1 + 2kl/D)^2} \tag{5.16}$$

The special discounting factor, $\beta \leq 1$, is required when the wake from the cluster C_1 does not cover the whole cluster C_2. To compute this discounting factor we assume that each position in the cluster has equal chance to have a turbine. We take the expectation of the interaction under this assumption. For example, if the wake covers only half of the cluster C_2 then it is likely that half of the turbines will be affected and the coefficient is 0.5. This coefficient also discounts the power penalty if not every turbine in the first cluster interacts with the second cluster. Formally, this coefficient is computed as:

$$\beta = \frac{1}{|C_2|} \sum_{i \in C_2} \sqrt{n_j / |C_1|} \tag{5.17}$$

where n_j is the number of the nodes in the cluster C_1 that interfere with the node j in the cluster C_2.

Note, that the interference between the clusters in (5.15) only depends on the number of the turbines in each cluster. We can now formulate the hierarchical optimization for wind farm planning.

5.3.3 Hierarchical optimization

The formula (5.15) gives us the power output of the cluster of the turbines. The unaffected power output of the cluster $P_0(C_k)$ is the solution of the within cluster maximization problem the power output only depends on the number of turbines installed in this cluster, z_k:

$$\begin{aligned} P_o(z_k) = \max_{\mathbf{x}} & \sum_i p x_i - \sum_{e_{ji} \in W} a_{ji} x_i x_j \\ \text{s.t.} & \sum_i x_i \leq z_k \\ & x_i \in \{0, 1\} \end{aligned} \tag{5.18}$$

Note, that the within cluster optimization uses the inferior linear superposition model. However, after the locations of the turbines in the cluster are chosen, we refine the power output with a more accurate model: in this work we use the quadratic superposition model. The objective of the layout optimization is the aggregate power output of all the clusters:

$$\max_{\mathbf{z}} \sum_{k \in Clusters} P_0(z_k) - \sum_{k,m \in Clusters} a_{mk} z_k \sqrt{z_m} \tag{5.19}$$
$$\text{s.t.} \sum_{k \in Clusters} z_k \leq N$$

The latter problem is the master optimization problem; the former is the child optimization problem. The number of the variables is not very large and we use the coordinate decent to optimize the master problem. The function $P_0(z_k)$ is computed empirically as a solution of the combinatorial child optimization problem, you can see the plot of this function in the figure 5.11. Minor optimization errors and combinatorial effects in the computation of $P_0(z_k)$ can lead to numerous local extrema of the master objective function (this actually happened in our experiments). To address this problem we approximate it with a quadratic polynomial. It allows us to compute the gradient with respect to z_k:

$$\frac{\partial P}{\partial z_k} = P_0'(z_k) - \sum_{m \in Clusters} a_{mk} \sqrt{z_m} - \sum_{m \in Clusters} a_{km} \frac{z_m}{2\sqrt{z_k}} \tag{5.20}$$

At each iteration we choose the variable with the highest gradient and the variable with the smallest gradient and reassign the turbines from the second cluster to the first cluster. If the budget constraint is not active we can simply add the turbines to the cluster with the highest gradient. Since only integral solutions are allowed, we can replace the gradient with function increments:

$$P_+' = P(z_k + 1) - P(z_k) \tag{5.21}$$
$$P_-' = P(z_k) - P(z_k - 1) \tag{5.22}$$

and add turbines to the cluster with the highest positive increment P_+' and remove them from the cluster with the lowest negative increment P_-'. The objective function may or may not be concave (depending on the coefficients). Multiple restarts and/or randomized steps can be beneficial. In our experiments non-concavity never created serious issues.

5.4 Results

We test our model on a 10km by 10km region segmented into a 10 by 10 grid. Each segment of a grid is a 1km by 1km region and represents the cluster of turbines optimized by a child

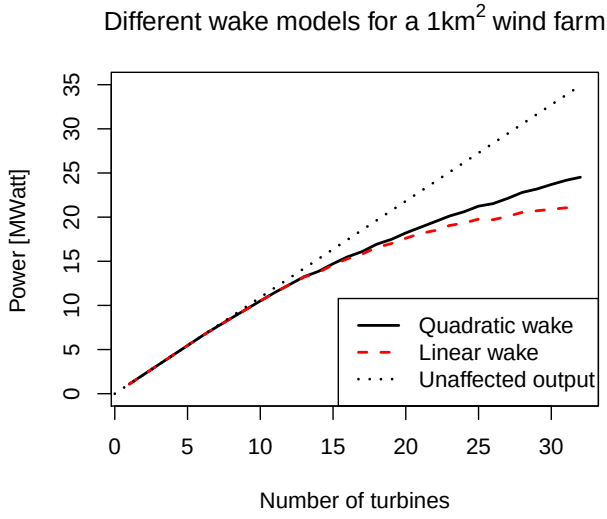


Figure 5.11: The power output computed with two different wake models: the linear model used in the baseline and a more accurate quadratic model used in our algorithm. The straight line shows potential unaffected output when there is no wake interference.

problem. Each cluster is in turn split into a 10 by 10 grid (100 meters between the nodes). That is, the farm has 10,000 candidate locations for the turbines. The wind speed varies uniformly in a [-45,+45] degrees sector. To integrate the objective function over varying wind speed directions we discretize the whole range into 100 intervals (less than 1 degree per interval). The turbines cannot be placed closer than 300 meters from each other. This is approximately four times the turbine diameter and is a standard guideline for the minimal distance.

We compare our hierarchical approach with the flat ILP baseline. Let us first report our findings concerning the behavior of the flat baseline. We will then report the results of our algorithm and compare it with the baseline, using the objective evaluation metrics.

Flat ILP baseline

The flat baseline has two major drawbacks: scalability and overestimation of the wake effect. In our experiments we have discovered that there are two very distinctive behaviors of the algorithm: when either the first or the second effect dominates. This mode depends on the combination of two factors: the size of the wind farm and the wake radius used in the model. Let us explain the nature of this effect. When the number of the turbines in the farm reaches certain density, the wake penalty for each additional turbine becomes greater than the power that this turbine

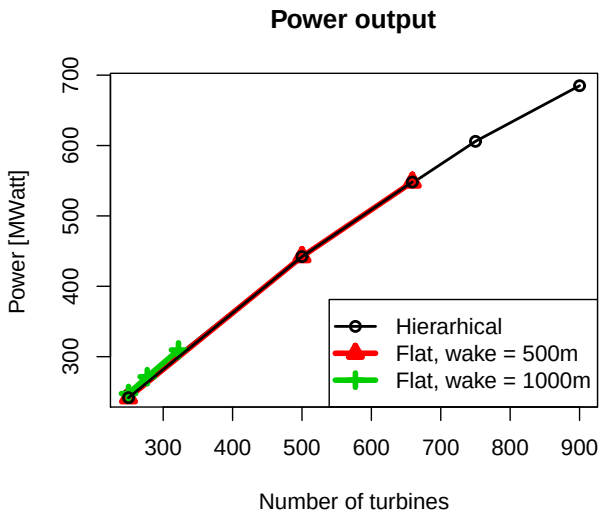
produces. Effectively, the wake model imposes a hard constraint on the smallest average distance between the turbines. This is true for any wake model, but the linear wake model used in the baseline overestimates the wake and this threshold distance becomes large. The larger the farm - the larger this threshold distance is. The difference in the wake models for a small farm is demonstrated in the figure 5.11. For larger farms the saturation of the power output occurs for much smaller density of the turbines. Practically, this threshold doesn't allow the optimizer to place the turbines too close to each other (on average). In the baseline solution this effect is partly compensated by the restricted wake radius. If the wake radius is smaller than the threshold, then *the wake radius essentially becomes a hard constraint in the baseline optimization.* If the wake radius is larger than the threshold, then the wake radius remains a soft constraint. Hard proximity constraints are much easier to satisfy, the problem becomes equivalent to packing as much balls in a box, as possible. The baseline ILP can solve this problem reasonably efficiently. The problem with soft constraints is much harder. We have already demonstrated it on the figure 5.6 in the section 5.2: the wake radius of 500 meters is the threshold point of that setup and it separates two behaviors of the algorithm.

The implication of this effect for the baseline is the following: the baseline either uses wake radius as a hard constraint or the problem becomes prohibitively expensive to solve. Our experiments confirm this observation. In the optimization of a 10km by 10km wind farm, the threshold wake radius is somewhere between 1 and 2 kilometers, i.e. at 1 km the problem is still a packing problem with hard proximity constraints and at 2km it becomes intractable.

Hierarchical optimization

Our hierarchical solution does not suffer from the problem above. In can place the turbines close to each other because it uses a more accurate wake model. Particularly, it models unlimited wake distance and does not suffer from wake overestimation. The power output is reported on the figure 5.12. When the number of the turbines is small, all the solutions perform equally well. However, as we explained above, the wake radius, that is less than 1km, acts as a hard constraint for the baseline, therefore it hits the maximum possible number of the turbines much sooner than our approach. Our hierarchical algorithm outperforms the baseline because it can find denser layouts. The timing shown in the right of the figure 5.12 demonstrates significant advantage of our approach. The times do not include the preprocessing steps, such as computation of interference coefficients.

Finally, let us demonstrate the qualitative difference between our solution and the baseline. Can it model the long-distance turbine interference effectively? The typical solutions are presented in the figure 5.13. The matrix on top of the figure shows the number of turbines per cluster, it characterizes the density of the turbines. The flat solution is not capable of modeling long-distance turbine interference and places all the turbines uniformly. The allocation pattern



Timing statistics

Method	Number of turbines	Timing [sec]
Hierarchical	250-1000	**0.02**
Flat, wake=500	250	110
	500	244
	685	1553
Flat, wake=1000	250	1867
	322	2332

Figure 5.12: Comparison of the power output functions (left) and the timing (right) of our hierarchical approach and the baseline.

of the hierarchical solution has a clear interpretation. The front rows of the farm where the wind is the strongest are denser. As we go from left to right, the wake effect accumulates and fewer turbines are installed to compensate for the power loss. The last row is dense again because we do not care about the wake outside the farm. The density on the sides is also higher. In the bottom of the figure the actual turbine placements are drawn. The rows in the adjacent clusters of our hierarchical solution are not aligned with each other and the allocation looks more chaotic. This happens because in the master optimization problem each cluster only knows the numbers of turbines in other clusters, but not the exact locations. Such behavior can lead to locally suboptimal positions, especially on the borders of the clusters. The local position adjustments will be able to solve this problem. This gives us the direction for the future work and the margin for the algorithm improvement.

To conclude, the hierarchical optimization method proposed in this chapter fully satisfies our expectations. It is capable of modeling long-distance turbine interference and greatly outperforms the baseline in both effectiveness and efficiency.

5.5 Summary

In this chapter we presented a hierarchical optimization framework for wind farm layout optimization. We purpose the following objectives. First of all, we demonstrate the advantages of the hierarchical decomposition in different domains. Second, we address the specific limitations of the existing wind farm layout optimization approaches: inability to model long distance turbine interference, exponential complexity and the use of the inferior interference wake model. We succeeded with all of these goals.

The hierarchical solution segments the location of the wind farm into multiple smaller regions. The master problem of the hierarchical solution decides how many turbines to install in each region. It takes into account the power output of each individual region and the interference between the regions. The optimization of the turbine placement within each region can be done independently of other regions. The power output of a region for a given number of turbines is computed using the accurate quadratic Jensen's wake model. The interference of any two regions is also computed with the quadratic model. The effects of several regions are combined using the linear superposition model. We do not restrict the range of the wake in our model. That is, we address the following limitations of the existing solutions: inability to model the long distance turbine interference and the use of the inferior linear model for superposition of the wakes.

Finally, the decomposition allows us to solve the problem very efficiently. We solve the master problem with the coordinate ascent. The optimization only takes a fraction of a second compared to hours or even days, required by the baseline solution. Most of the time is spent on preprocessing and on solving the children allocation problems (we use the baseline solution to do that).

Our major goal for the future work is to design an optimization algorithm to fine-tune the solution. When the two adjacent regions are glued together in the global farm design, there may appear locally suboptimal turbine placements, especially on the boundaries of the regions. Addressing this issue will improve the power output and can potentially improve the visual attractiveness of the wind farm.

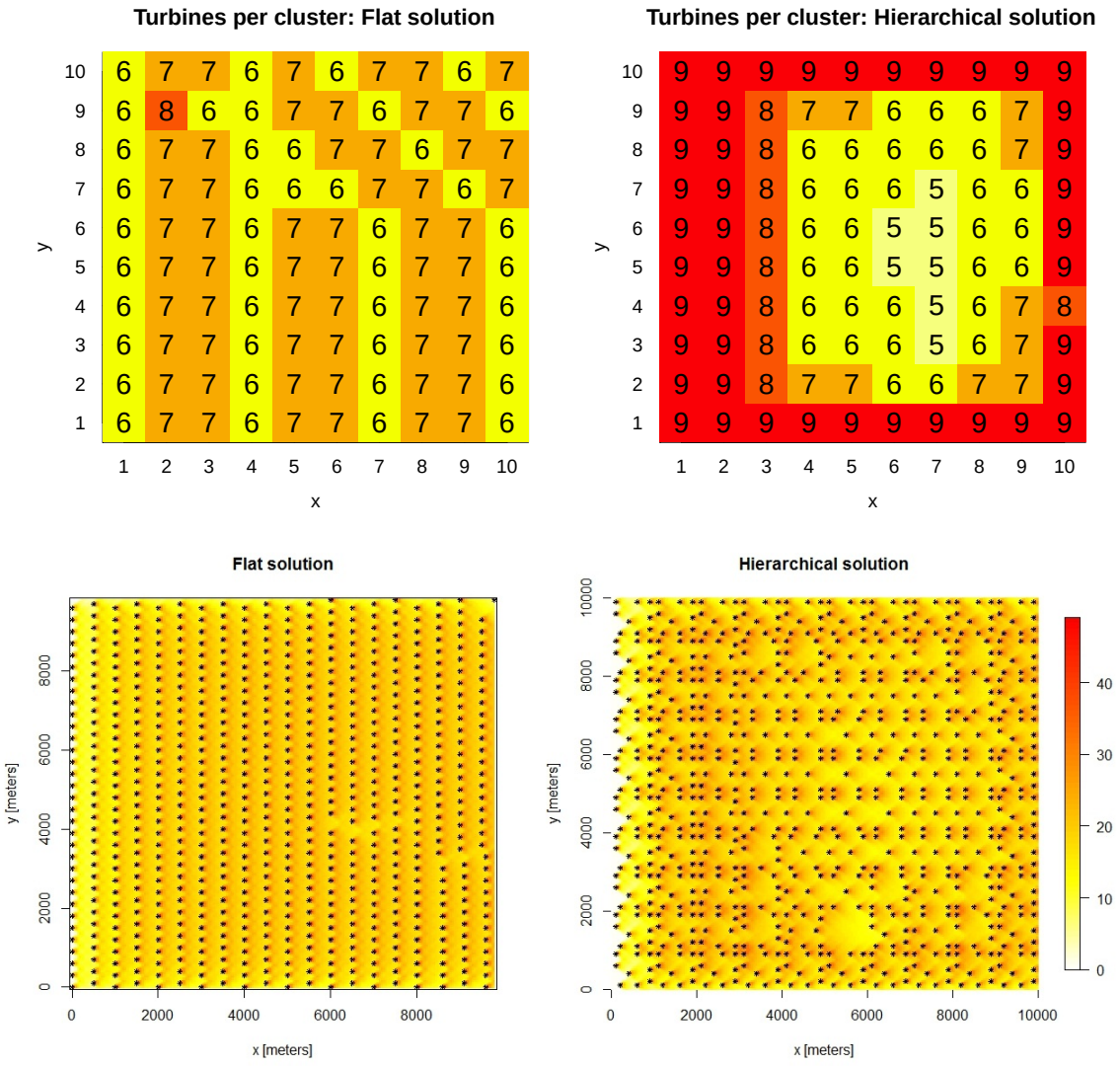


Figure 5.13: **Top**, the number of turbines per cluster of typical flat and hierarchical solutions. **Bottom**, the positions of the turbines and the wind deficit map.

Bibliography

[1] Zoe Abrams, Ofer Mendelevitch, and John Tomlin. Optimal delivery of sponsored search advertisements subject to budget constraints. In *Proceedings of the 8th ACM conference on Electronic commerce*, EC '07, pages 272–278, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-653-0. doi: http://doi.acm.org/10.1145/1250910.1250950. URL http://doi.acm.org/10.1145/1250910.1250950. 2.1, 2.1, 2.4.1, 2.7.1

[2] Manda Adams and David Keith. A wind farm parameterization for wrf. In *Proceedings of the 8th WRF users workshop*, 2007. (document), 5.2, 5.2.1

[3] Deepak Agarwal, Datong Chen, Long-ji Lin, Jayavel Shanmugasundaram, and Erik Vee. Forecasting high-dimensional data. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 1003–1012, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0032-2. doi: 10.1145/1807167.1807277. URL http://doi.acm.org/10.1145/1807167.1807277. 2.3

[4] G. Anandalingam and T. L. Friesz. Hierarchical optimization: an introduction. *Ann. Oper. Res.*, 34:1–11, February 1992. ISSN 0254-5330. doi: http://dx.doi.org/10.1007/BF02098169. URL http://dx.doi.org/10.1007/BF02098169. 3.2.2

[5] World Wind Energy Association. World wind energy report 2010, 2011. URL http://www.wwindea.org/home/images/stories/pdfs/worldwindenergyreport2010_s.pdf. 5.1

[6] R.E. Bellman. *Dynamic Programming*. Dover Books on Mathematics. Dover Publications, 2003. ISBN 9780486428093. URL http://books.google.com/books?id=fyVtp3EMxasC. 2.4.3, 2.4.3, 2.4.3

[7] Michael Benisch, Norman Sadeh, and Tuomas Sandholm. Methodology for designing reasonably expressive mechanisms with application to ad auctions. In *In International Joint Conference on Artificial Intelligence*, 2009. 2

[8] BloombergBusinessweek. http://www.businessweek.com/news/2012-02-07/wind-power-market-rose-to-41-gigawatts-in-2011-led-by-china.html, 2012. 5.1

[9] Craig Boutilier, David C. Parkes, Tuomas Sandholm, and William E. Walsh. Expressive

banner ad auctions and model-based online optimization for clearing. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 1*, AAAI'08, pages 30–37. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1619995.1620002. 2.1

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787. 2.6

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787. 3.2.3

[12] Jerome Bracken and James T. McGill. Mathematical Programs with Optimization Problems in the Constraints. *Operations Research*, 21:37–44, 1973. 3.2.2

[13] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th annual European conference on Algorithms*, ESA'07, pages 253–264, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-75519-5, 978-3-540-75519-7. URL http://dl.acm.org/citation.cfm?id=1778580.1778606. 2.6

[14] P. H. Calamai and J. J. Moré. Projected gradient methods for linearly constrained problems. *Mathematical Programming*, 39:93–116, 1987. URL http://www.ams.org/mathscinet-getitem?mr=89f:90132. 3.1, 3.3

[15] Denis Charles, Deeparnab Chakrabarty, Max Chickering, Nikhil R. Devanur, and Lei Wang. Budget smoothing for internet ad auctions: a game theoretic approach. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, EC '13, pages 163–180, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1962-1. doi: 10.1145/2482540.2482583. URL http://doi.acm.org/10.1145/2482540.2482583. 2.1

[16] H. Cheng and E. Cantú-Paz. Personalized click prediction in sponsored search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 351–360, 2010. 2.1, 2.2

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, second edition, 2001. 2.4.2

[18] A. A. Cournot. *Researches into the Mathematical Principles of the Theory of Wealth*. Translated in 1897 by N.T. Bacon, with a bibliography of Mathematical Economics by Irving Fisher. The Macmillan Company, New York, 1838. 4.2

[19] G.B. Dantzig. *Linear programming and extensions*. Landmarks in Physics and Mathematics. Princeton University Press, 1998. ISBN 9780691059136. URL http://books.google.com/books?id=2j46uCX5ZAYC. 2.3

[20] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Opera-*

tions Research, 8(1):pp. 101–111, 1960. ISSN 0030364X. URL http://www.jstor.org/stable/167547. 3.1, 3.4

[21] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 71–78, New York, NY, USA, 2006. ACM. ISBN 1-59593-134-1. doi: 10.1145/1132516.1132527. URL http://doi.acm.org/10.1145/1132516.1132527. 4.1, 4.2.5

[22] Nikhil R. Devanur, Bach Q. Ha, and Jason D. Hartline. Prior-free auctions for budgeted agents. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, EC '13, pages 287–304, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1962-1. doi: 10.1145/2482540.2482554. URL http://doi.acm.org/10.1145/2482540.2482554. 2.1

[23] Nikhil R. Devenur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pages 71–78, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-458-4. doi: http://doi.acm.org/10.1145/1566374.1566384. URL http://doi.acm.org/10.1145/1566374.1566384. 2.1, 2.6, 2.6

[24] S. Donovan. Wind farm optimization. *University of Auckland, Dept. Engineering Science*, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.8323&rep=rep1&type=pdf. 5.1, 5.2.4

[25] B. Edelman and M. Schwarz. Optimal auction design and equilibrium selection in sponsored search auctions. *American Economic Review Papers and Proceedings*, 100(2):597–602, 2010. 2.1, 4.1

[26] 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*, 97:242–259, 2007. 2.1, 2.2, 4.1

[27] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004. 4.4.1

[28] Christopher N Elkinton, James F Manwell, and Jon G McGowan. Offshore wind farm layout optimization (owflo) project: an introduction. *Offshore Wind*, pages 1–9, 2005. URL http://wind.nrel.gov/public/SeaCon/Proceedings/Copenhagen.Offshore.Wind.2005/documents/papers/Poster/C.Elkinton_Offshore_WindFarmLayoutOptimization_OWFLO.pdf. 5.1, 5.2.2

[29] Patrik Fagerfjall. Optimizing wind farm layout more bang for the buck using mixed integer linear programming. Master's thesis, Chalmers University of Technology and Gothenburg

University, Goteborg, Sweden, 2010. 5.1, 1, 5.2.4, 5.3.2

[30] Jon Feldman, Nitish Korula, Vahab Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *Proceedings of the 5th International Workshop on Internet and Network Economics*, WINE '09, pages 374–385, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10840-2. doi: 10.1007/978-3-642-10841-9_34. URL http://dx.doi.org/10.1007/978-3-642-10841-9_34. 2.1, 2.6

[31] LLC Global Energy Concepts. Sandia National Laboratories, New Mexico wind resource assessment, Lee ranch, 2003. URL http://windpower.sandia.gov/other/LeeRanchData-2002.pdf. (document), 5.3

[32] T. Graepel, J.Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proc. 27th International Conference on Machine Learning*, 2010. 2.1, 2.2

[33] Qi Guo and Eugene Agichtein. *Ready to buy or just browsing?: detecting web searcher goals from interaction data*. SIGIR '10. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0153-4. doi: 10.1145/1835449.1835473. URL http://doi.acm.org/10.1145/1835449.1835473. 3.1

[34] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):pp. 100–108, 1979. ISSN 00359254. URL http://www.jstor.org/stable/2346830. 3.1, 3.5

[35] IAB. http://www.iab.net/adrevenuereport, 2012. 2.1

[36] N. O. Jensen. A note on wind generator interaction, 1984. URL http://130.226.56.153/rispubl/VEA/veapdf/ris-m-2411.pdf. 5.2.3

[37] Shizuo Kakutani. A generalization of Brouwer's fixed point theorem. *Duke Mathematical Journal*, 8:416–427, 1941. URL http://projecteuclid.org/Dienst/UI/1.0/Summarize/euclid.dmj/1077492791?abstract=. 4.2.2

[38] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972. 4.3.3, 4.3.3

[39] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC '90, pages 352–358, New York, NY, USA, 1990. ACM. ISBN 0-89791-361-2. doi: http://doi.acm.org/10.1145/100216.100262. URL http://doi.acm.org/10.1145/100216.100262. 2.1

[40] I. Katic, J. Højstrup, and N.O. Jensen. A simple model for cluster efficiency. In W. Palz and E. Sesto, editors, *EWEC'86. Proceedings. Vol. 1*, pages 407–410, 1987. 5.2.3

[41] Sungchul Kim, Tao Qin, Hwanjo Yu, and Tie-Yan Liu. An advertiser-centric approach to understand user click behavior in sponsored search. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management*, 2011. 2.1, 2.2

[42] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983. 4.3.3

[43] Christian Kroer and Tuomas Sandholm. Computational Bundling for Auctions. Technical Report CMU-CS-13-111, Carnegie Mellon University, School of Computer Science, 05 2013. URL `http://www.cs.cmu.edu/~sandholm/bundling_tech_report.pdf`. 3.1

[44] Chris Lake. http://econsultancy.com/us/blog/7345-how-to-use-paid-search-as-a-branding-tool, 2011. 2.1

[45] C. E. Lemke and Howson, Jr. Equilibrium Points of Bimatrix Games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423, 1964. 4.2.5

[46] Marco Lubbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2004. 2.1, 2.4.2

[47] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized on-line matching. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 264–273, 2005. doi: 10.1109/SFCS.2005.12. 2.6

[48] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*, pages 65–77. Oxford University Press, January 2002. ISBN 0-19-512594-0. URL `http://www.rpi.edu/~mitchj/papers/bc_annot.html`. 4.3.3

[49] G. Mosetti, C. Poloni, and B. Diviacco. Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics*, 51(1):105 – 116, 1994. ISSN 0167-6105. doi: http://dx.doi.org/10.1016/0167-6105(94)90080-9. URL `http://www.sciencedirect.com/science/article/pii/0167610594900809`. 5.1

[50] V. Nanduri and T. Das. A reinforcement learning algorithm for obtaining the Nash equilibrium of multi-player matrix games. *IIE Transactions*, 41(2):158–167, 2009. 4.2.5, 4.4.3

[51] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):pp. 286–295, 1951. ISSN 0003486X. URL `http://www.jstor.org/stable/1969529`. 4.2, 4.2.2, 4.2.5

[52] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007. ISBN 0521872820. 4.2, 4.2.3, 4.2.5

[53] Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. The MIT Press, Cambridge, USA, 1994. ISBN 0-262-65040-1. electronic edition. 4.1

[54] Michael Ostrovsky and Michael Schwarz. Reserve prices in internet advertising auctions: a field experiment. In *Proceedings of the 12th ACM conference on Electronic commerce*, EC '11, pages 59–60, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0261-6. doi: 10.1145/1993574.1993585. URL `http://doi.acm.org/10.1145/1993574.1993585`. 2.2

[55] Pamela Parker. www.clickz.com/clickz/news/1694929/ sempo-most-search-advertisers-want-brand-impact, 2004. 2.1

[56] David C. Parkes and Tuomas Sandholm. Optimize-and-Dispatch Architecture for Expressive Ad Auctions. In *Proceedings of First Workshop on Sponsored Search Auctions*, 2005. 2.1

[57] Pittsburgh Post-Gazette. http://www.post-gazette.com/stories/local/breaking/without-wind-pines-turbine-is-a-bust-282451/, 2011. 5.1

[58] Bem Roberts, Dinan Gunawardena, Ian A. Kash, and Peter Key. Ranking and tradeoffs in sponsored search auctions. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, EC '13, pages 751–766, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1962-1. doi: 10.1145/2482540.2482568. URL `http://doi.acm.org/10.1145/2482540.2482568`. 2.1

[59] Stephen M. Robinson. A characterization of stability in linear programming. *Operations Research*, 25(3):pp. 435–447, 1977. ISSN 0030364X. URL `http://www.jstor.org/stable/169931`. 3.5

[60] Michele Samorani. The wind farm layout optimization problem. *Leeds School of Business Research Paper Series, University of Colorado at Boulder*, 2010. 5.1, 5.2.4

[61] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. 4.3.2

[62] N. Z. Shor, Krzysztof C. Kiwiel, and Andrzej Ruszcayǹski. *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc., New York, NY, USA, 1985. ISBN 0-387-12763-1. 3.1, 3.3

[63] Thomas Sorensen, M. Lybech Thogersen, Per Nielsen, and EMD International A/S. Adapting and calibration of existing wake models to meet the conditions inside offshore wind farms. 2008. 5.2.3, 5.2.3

[64] David R.M. Thompson and Kevin Leyton-Brown. Revenue optimization in the generalized second-price auction. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, EC '13, pages 837–852, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-

1962-1. doi: 10.1145/2482540.2482612. URL http://doi.acm.org/10.1145/2482540.2482612. 2.1

[65] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994. 4.1, 11, 4.4.1

[66] G. van der Laan, A. J. J. Talman, and L. van der Heyden. Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. *Math. Oper. Res.*, 12(3):377–397, August 1987. ISSN 0364-765X. doi: 10.1287/moor.12.3.377. URL http://dx.doi.org/10.1287/moor.12.3.377. 4.2.5

[67] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, pages 109–118, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-822-3. doi: http://doi.acm.org/10.1145/1807342.1807360. URL http://doi.acm.org/10.1145/1807342.1807360. 2.1, 2.6

[68] William Walsh, Craig Boutilier, Tuomas Sandholm, Rob Shields, George Nemhauser, and David Parkes. Automated channel abstraction for advertising auctions, 2010. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1905. 2.1, 2.4, 3.1, 3.7

[69] M. P. Wand and M. C. Jones. *Kernel Smoothing (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 1 edition, December 1994. ISBN 0412552701. URL http://www.worldcat.org/isbn/0412552701. 2.7.2

[70] Wallodi Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18:293–297, 1951. 5.2.2

[71] Lori Weiman. http://searchengineland.com/top-5-ways-to-use-search-for-branding-66332, 2011. 2.1

[72] Wikipedia. http://en.wikipedia.org/wiki/online_advertising, 2012. 2.1

[73] Chenyan Xiong, Taifeng Wang, Wenkui Ding, Yidong Shen, and Tie-Yan Liu. Relational click prediction for sponsored search. In *Proceedings of the fifth ACM Conference on Web Search and Data Mining*, 2012. 2.1, 2.2

[74] Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 25–32, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3. doi: 10.1145/502585.502591. URL http://doi.acm.org/10.1145/502585.502591. 3.7