# PRIVACY PROTECTION AND ADVERTISING IN A NETWORKED WORLD

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Gagan Aggarwal
September 2005

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Rajeev Motwani Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Boneh

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ashish Goel

Approved for the University Committee on Graduate Studies.

# Abstract

The last couple of decades have witnessed a phenomenal growth in the networking infrastructure connecting computers all over the world. The Internet has now become an ubiquitous channel for information sharing and dissemination. This has created a whole new set of research challenges, while giving a new spin to some existing ones. In this thesis, we address problems from two areas: (a) protection of data privacy, and (b) sale of advertisement space on Internet web sites.

The first part of the thesis deals with privacy issues involved in the exchange of information between non-trusting entities. The scenarios of interest include the Census Bureau publishing population statistics, hospitals sharing patient data with medical researchers, federal agencies sharing intelligence information with each other, a group of people wishing to find their common interests, and several universities trying to compute combined statistics about faculty salaries, among others. In most cases, exchange of a relevant synopsis or aggregate would be sufficient; however, in the absence of knowledge about what synopsis would be most relevant, the data tends to be disseminated in the raw. This threatens personal privacy and creates an opportunity for dishonest entities to misuse the information to further their own selfish agenda. We focus our attention on two abstract problems derived from this setting. We first consider the problem of anonymizing databases before dissemination, so as to safeguard the privacy of the individuals described by the databases. A solution to this problem can be used by the Census Bureau as well as hospitals to provide data containing personal information to interested parties without sacrificing privacy. We adopt the privacy framework of $k$-anonymity proposed by Samarati and Sweeney, and present approximation algorithms for anonymizing databases. The second problem we study is that of computing a function over data split between two or more non-trusting entities. The goal is to enable

the entities to compute the value of the function without either entity having to reveal any unnecessary information to the other entity. In particular, we study the problem of computing statistics over data shared between multiple entities. We present efficient protocols for computing a fundamental statistical function, namely the $k$-th ranked element of a dataset split between multiple entities. These protocols form a practical solution to the problem of compiling faculty salary statistics covering several universities.

The pervasiveness of the Internet has fueled a rapid growth in advertising on Internet web sites. In the second part of the thesis, we consider the problem of selling advertisement space on the Internet. Given the dynamic nature of the online advertising market – advertisers join and leave, the popularity of the web site changes over time, the value of an Internet user clicking on an advertisement link varies over time – auctions are the logical choice for selling advertisement space on web sites, and indeed, major search engines like Google and Yahoo! are using auctions to sell advertisement slots on their search result pages. However, a lack of understanding of good bidding strategies has kept marketers from fully embracing online advertising channels. One solution is to use selling mechanisms where the best strategy for advertisers is simple and well-understood. The class of truthful mechanisms has the property that the best strategy for any advertiser is to bid an amount equal to her true valuation of the object she is bidding for. Since the use of truthful auction mechanisms considerably simplifies the task of bidding, we propose using truthful auctions for selling web advertisements. We study two different problem formulations in this setting. We first consider the problem of selling a single slot on a web page that gets a known number of hits per day, assuming that the number of visitors desired by each advertiser is known in advance. We present a truthful auction that is competitive with respect to an optimal *omniscient* pricing scheme that obeys a natural monotonicity property. The second problem we study is that of selling multiple advertisement slots on a web page. This problem is more closely aligned with the problem faced by search engines. We present an auction that is truthful when the advertisers are not budget-constrained. Moreover, under some reasonable assumptions, we show that its revenue is equivalent to the non-truthful auctions currently being used by search engines.

# Acknowledgements

Many people have helped me during my stay at Stanford, and in the years leading upto it. To all of them, I owe my thanks.

First, I wish to express my gratitude to the faculty of the Department of Computer Science and Engineering at the Indian Institute of Technology, Delhi, for providing me with an excellent undergraduate experience and a solid academic foundation.

I owe my sincerest thanks to my advisor, Rajeev Motwani, for his invaluable advice and encouragement over the years. He has been an outstanding advisor — always very flexible and willing to let his students work at their own pace, while making sure that things are going alright. He is an inexhaustible source of research problems that are both interesting and well-motivated at the same time. I always found him highly accessible and I thank him for all the weekly meetings that kept me motivated.

I would like to thank my thesis readers, Dan Boneh and Ashish Goel, for their encouragement and suggestions related to further research directions. I thank Hector Garcia-Molina who helped me appreciate the power of "examples" in conveying an idea. I would like to thank Benny Pinkas for introducing me to the area of secure function evaluation and privacy-preserving computation, and for being a great mentor. I also wish to thank Suresh, Bala, David, Sridhar, Jayram, Nina and Jason, who mentored me during my summer internships. I would like to thank Moses, Sudipto, Piotr and Sanjeev Khanna for their support and encouragement.

The best thing about my PhD at Stanford was the opportunity to do research with extremely bright and energetic students and other colleagues. I would like to thank An, Tomas, Rina, Devavrat, Mayur, Brian, Krishnaram, Dilys, Mayank, Prasanna, and all the others who made research at Stanford an extremely enjoyable experience. A lot of the

research that appears in this thesis is joint work with them.

The theory wing was a fun place to be, and I would like to thank my wingmates, especially An, Liadan, Mayur, Aris, Kamesh, Adam, Gurmeet, Ben, Aditi, Sergei and Ying, for never being too busy for a chat or a refreshing cup of tea. An even let me take the window space in our office :) I thank Lynda, Maggie and Sondra for "reminding" me to submit my reimbursement requests (otherwise, I might have been an even poorer graduate student), and for making our wing a more welcoming place.

My five years at Stanford have been a wonderful experience, and I thank all my friends who made it so. I owe my special thanks to Vibha, my roommate for the early part of my PhD, whose company made the process of adapting to a new place effortless. I will always cherish the time that we spent together. I would like to thank Manali for organizing all those get-togethers that helped me keep in touch with friends. I thank Matthias for the moral and technical support, and for lending his cheerful company for many a movie and dinner. I thank Pooja (and her Honda CRV) and Manali for their generous help with groceries and airport pick-ups when I did not own a car. I have received a lot of earnest advice from my cousin Vishal about various aspects of living in the United States, and I thank him for that. I would also like to thank Mitali, Pooja, Madhu, Sachin, and many others for their friendship and support.

Most of all, I would like to thank my family — my brother Omesh with whom I have shared many a memorable moment, and my parents who always believed in me, and gave me unconditional love, support and encouragement at every step of the way. For their love and for everything else, I dedicate this thesis to them.

Gagan Aggarwal
Stanford, California
September 15, 2005

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The last couple of decades have witnessed a phenomenal growth in the networking infrastructure connecting computers all over the world. The Internet has now become an ubiquitous channel for information sharing and dissemination. This has created a whole new set of research challenges, while giving a new spin to some existing ones. In this thesis, we address problems from two areas: (a) protection of data privacy, and (b) sale of advertisement space on Internet web sites.

## 1.1 Privacy in a Networked World

The first part of the thesis deals with privacy issues involved in the exchange of information between non-trusting entities. The scenarios of interest include the Census Bureau publishing population statistics, hospitals sharing patient data with medical researchers, federal agencies sharing intelligence information with each other, a group of people wishing to find their common interests, and several universities trying to compute statistics about faculty salaries, among others. In most cases, exchange of a relevant synopsis or aggregate would be sufficient; however, in the absence of knowledge about what synopsis would be most relevant, the data tends to be disseminated in the raw. This threatens personal privacy and creates an opportunity for dishonest entities to misuse the information to further their own selfish agenda. We focus our attention on two facets of this problem.

### 1.1.1 Anonymization

More and more data is being exchanged and even posted on the Internet without sufficient anonymization. Moreover, one can use increasingly sophisticated tools to integrate data from different sources and reconstruct highly detailed information about an individual. One way to deal with the threat to rising threat to personal privacy is to refuse to divulge any private information. However, in many scenarios, information exchange can prove socially beneficial. For example, if medical researchers have access to databases containing the medical histories of various individuals, they can discover the association between certain lifestyle factors and higher risk of certain diseases; geographical occurrence data on communicable diseases can enable detection of the outbreak of epidemics at an early stage, thereby preventing its spread to larger populations. With the goal of enabling such applications, it is quite desirable that hospitals make their records available to medical scientists. At the same time, such personal data has a great potential for misuse; for example, a health insurance company could exploit such data to selectively raise the health insurance premiums of certain individuals. Unfortunately, we cannot trust all medical researchers to uphold the privacy of the data. Even if they are all honest and well-meaning, they might not be technically savvy enough to prevent unauthorized access.

One possible solution is that instead of releasing the entire database, the database owner answers aggregate queries posed by medical researchers after ensuring that answers to the queries do not reveal sensitive information. This approach is called query auditing [KPR03, KMN05, DN04a]. This requires the researchers to formulate their queries without access to any data. In this case, one can also use techniques from *secure multi-party computation* [Yao86, GMW87, LP02, AMP04, FNP04]. However, many of the data-mining tasks are inherently ad hoc and the data mining researchers need to examine the data in order to discover data aggregation queries of interest. In such cases, query auditing and secure function evaluation techniques do not provide an adequate solution, and we need to release an anonymized view of the database that enables the computation of non-sensitive query aggregates, perhaps with some error or uncertainty.

We consider the problem of anonymizing databases before dissemination, in order to safeguard the privacy of the individuals described by the databases. One approach to

anonymization is to use *perturbation* techniques in order to hide the exact values of the data [AS00, AA01, DN03, DN04b, EGS03, AST05, CDM+05]. However, this may not be suitable if one wants to draw inferences with 100% confidence. Another approach is to *suppress* some of the data values, while releasing the remaining data values exactly. We note that suppressing just the identifying attributes, like name and address, is not sufficient to protect privacy. In order to ensure privacy, we adopt the privacy framework of $k$-Anonymity which was proposed by Samarati and Sweeney [Swe02, SS98]. Suppose we have a table with each tuple having $m$ quasi-identifying attributes. The $k$-Anonymity framework provides for suppressing or generalizing (see Chapter 2) some of the entries in the table so as to ensure that *for each tuple in the modified table, there are at least* $k-1$ *other tuples in the modified table identical to it.* The idea is that even if an adversary gets hold of all the quasi-identifying attributes of all the individuals in the table, it cannot track down an individual's record further than a set of $k$ records in the worst case. We study the problem of making a database $k$-Anonymous, while minimizing the extent of suppression/generalization and provide approximation algorithms for it. The algorithms and hardness results (joint work with Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas and An Zhu) were originally published in [AFK+05] and are described in Chapter 2.

### 1.1.2 Secure Computation

The ease of transferring data over computer networks has led to an increase in data exchange and sharing. Many a times, two entities who do not know or trust each other trade information for mutual benefit. For example, two corporations might wish to exchange information about their respective customer bases, in order to evaluate the viability of a merger; two or more federal agencies might wish to exchange intelligence information with each other; a group of people might wish to find out their common interests; a set of universities might try to compile statistics on faculty salaries (the Taulbee survey), ans so on. In many of these cases, the entities are interested in sharing some kind of aggregate information about private data that each of them holds. Examples of aggregation functions

include statistics over data sets, clustering over data points, intersection (size) of sets, Hamming distance between vectors etc. While the parties feel comfortable sharing aggregate information, typically they do not wish to reveal the raw data to each other. In such a case, it is desirable to perform the computation of the aggregate information without having to reveal *any* information that is not implied by the aggregate itself. One might argue that this is an unnecessarily stringent requirement and harmless information leaks should be permitted; however, in the absence of a good characterization of *harmless* leaks, we will insist on not leaking any additional information. This privacy requirement has been formalized in the cryptography literature as *Secure Function Evaluation (SFE)* against a *semi-honest* or a *malicious* adversary. Let $x$ be the private data held by one entity and $y$ be the private data held by the other entity. It has been shown [Yao86, GMW87] that it possible to evaluate any function $f(x, y)$ that can be represented as a Boolean circuit $C(x, y)$ using a protocol that has a communication and computation overhead of $O(|C(x, y)|)$. While such an overhead is acceptable for simple functions and small data sizes, these protocols are prohibitively expensive for computing functions over large databases or computing complicated functions represented by large circuits. In this case, it is desirable to have special-purpose protocols to compute specific functions more efficiently. In fact, efficient protocols have recently been developed for computing a decision tree for classification securely using an approximate version of the ID3 algorithm [LP02], secure computation of approximate Hamming distance between two vectors held by different parties [FIM$^+$01], and secure computation of the intersection of sets held by different parties [FNP04], among others.

We study the problem of computing a basic statistical quantity, namely the $k^{th}$ ranked element of a set shared between two or more parties in a privacy-preserving manner. Of particular interest is the median of a set shared by two or more parties. First consider the following application scenario. Two health insurance companies wish to compute the median life expectancy of the smokers insured by them. In this setting, both the number of insured smokers as well as their life expectancies are private information, but the median life expectancy is of combined mutual interest. Another example is the annual Taulbee survey which collects salary and demographic data for faculty in computer science and computer engineering departments in North America. Typically, academic departments report only a small number of statistics like the minimum, maximum, average and median

salary of professors. The Taulbee survey is thus able to publish only limited aggregate information. An efficient and secure multi-party solution for computation of the $k^{th}$-ranked element would enable universities to quickly compute combined salary statistics without revealing individual salary values to others. It would even facilitate secure computation of salary histograms [GMP97, PGI99, JKM+98].

We take the same approach as that of previous solutions for secure computation over large inputs (e.g. [LP02, FIM+01, CIK+01]), and reduce this task to many invocations of secure computation of simpler functions over small inputs; but unlike these constructions, we also design protocols which are secure against malicious adversaries. We present the protocols and proofs of their security (joint work with Nina Mishra and Benny Pinkas and originally published in [AMP04]) in Chapter 3.

## 1.2 Auctions for Web Advertisements

The development of excellent networking infrastructure has made the Internet a pervasive influence in people's lives. Marketers have responded by pushing more of their budgets online, especially into search advertising, display ads, and rich-media TV-style ads. Search engines like Google and Yahoo!, most of whose profits come from advertising on the Internet, are showing a rapid increase in profits [Goo05, Yah05]. According to a forecast from Forrester Inc. [For05], in 2010, marketers will spend $26 billion on online advertising, which will represent 8% of all advertising spending, rivaling spending on cable/satellite TV and radio.

The web advertisement market is highly dynamic in nature, with advertisers arriving and leaving all the time. Moreover, the number of hits received by a website changes over time, as does the value of an Internet user clicking on an advertisement. This makes auctions the logical choice for selling advertisement space on web sites. Indeed, major search engines like Google and Yahoo! are using auctions to sell advertisement slots on their search result pages. However, none of the existing auctions provides a clear-cut best strategy for bidding. Thus, an advertiser must understand the underlying game-theoretic issues in order to be able to bid well. In fact, according to the marketers surveyed by Forrester [For05], "a lack of online advertising standards and hands-on experience have

kept marketers from fully embracing online channels". One possible way to attract more advertisers to the online advertising marketing is to use selling mechanisms where the best strategy for advertisers is simple and well-understood. The class of truthful mechanisms has the property that the best strategy for a bidder is to bid an amount equal to their true valuation (i.e., an estimate of worth) of the object they are bidding for. In scenarios where this true valuation is known (or can be evaluated), the use of truthful auction mechanisms simplifies the task of bidding immensely. With this consideration in mind, we wish to develop truthful auctions for selling web advertisements. We further discuss the desirability of using truthful auctions in Chapter 4.

Consider a web page with some slots where advertisements can be displayed. Whenever an Internet user accesses the web page, the web page owner can choose to display one or more advertisements. The process of displaying an advertisement is called an *impression*. Depending on the content of the web page, a variety of advertisers might be interested in displaying advertisements on the web page. Each of these advertisers will invoke a different level of interest from the Internet users visiting the web page. Thus, each advertiser will have a different click-through rate (CTR) associated with her advertisement (the click-through rate of an advertisement is the fraction of its impressions that result in a click by an Internet user). We assume that the web page owner has (or can collect) statistical information about the CTRs of various advertisers. Furthermore, each advertiser values an impression or a click on her advertisement differently and this valuation is known only to her. The web page owner wants to design a selling mechanism for the advertisement slot(s) on her web page with the goal of maximizing profit.

The problem of designing a good auction for this setting is multi-faceted — advertisers may have combinatorial preferences (when multiple keyword combinations are relevant to an advertisement campaign), advertisers may have restricted funds (limited budget) to spend on advertising, the search engines might (and usually do) display multiple advertisements for every search keyword, and availability and needs may vary over time. Each of these aspects of the problem in themselves represents a significant auction design challenge. In this thesis, we focus on two particular, yet fundamental, problem formulations in this setting.

### 1.2.1 Selling a Single Advertisement Slot

We first study the problem of selling a single advertisement spot on a web page over the course of a day. We assume that the web page owner knows the (average) number of times the web page is accessed in a day. We let each advertiser have a limit on the number of clicks on her advertisement that she can handle in a single day. We assume that these limits are known to the web page owner, perhaps due to repeated interaction with the advertisers. The web page owner can use these limits together with the click-through rates of various advertisements, and compute the maximum number of impressions desired by each advertiser. This will be referred to as the *demand* of the advertiser. It is assumed that the valuation of a click to an advertiser is not known to the auctioneer. We wish to develop a truthful auction for this problem with the goal of maximizing revenue.

The above problem can be modeled as a *private-value* version of the (fractional) *knapsack problem* (the limitation imposed by the number of times a web page is accessed in a day acts as the capacity of the knapsack; details are given in Chapter 5). We will refer to this version as the *knapsack auction problem*. It models several other interesting applications as well. For example, knapsack auctions can be used to model auctions for satellite bandwidth. Suppose a satellite broadcasting service provider has a total bandwidth of $C$ and content providers have different bandwidth needs, i.e., $c_i$ for provider $i$, and different valuations for the fulfillment of their needs. This problem translates directly into the knapsack auction problem assuming that it is not possible for providers to falsely declare their bandwidth needs.

Through the study of the knapsack auction problem, we wish to develop a better understanding of how to do prior-free optimization (i.e. optimization without having any prior knowledge about the distribution of the valuations of various agents) when there are non-trivial constraints on the allocation. In our case, items selected for the knapsack must all fit in the space available. In addition to presenting a knapsack auction that performs well (discussed next), we outline a general approach for dealing with non-trivial optimization problems. The first step of this approach is to solve the *unlimited-supply* version of the problem. The second step is to select a suitable subset of the bidders and simulate the

unlimited supply auction on this subset. This general approach works for *monotone* optimization problems, where if an allocation is feasible, then any subset of the allocation is also feasible.

Following Goldberg et al. [GHW01], we analyze knapsack auctions in the framework of *competitive analysis* by comparing the performance of the auction to an *optimal omniscient pricing*. We consider pricing rules that obey the following natural constraint: bidders desiring more capacity should not be offered lower prices than those desiring less. We refer to this as *monotone pricing*, since the valid pricing functions for this class are monotone non-decreasing. Accordingly, we define $\mathrm{OPT}$ to be the profit obtained by the best monotone pricing function for bidders' actual valuations when we assume that a bidder pays the offered price if and only if it is no more than their valuation. Because it is not possible to obtain a constant fraction of $\mathrm{OPT}$ in the worst case, we design auctions that obtain at least a constant fraction of $\mathrm{OPT}$ less a small additive loss term, i.e., $\alpha\,\mathrm{OPT} - \lambda h$ (where $h$ is an upper bound on the highest bidder's valuation). Ideally, we would like both $\alpha$ and $\lambda$ to be constants. We present an auction that achieves a constant $\alpha$ and $\lambda \in O(\log\log\log n)$. Further details of the competitive analysis framework can be found in Chapter 4. The auction is presented and analyzed in Chapter 5. These results are joint work with Jason Hartline and will appear in [AH06].

### 1.2.2 Selling Multiple Advertisement Slots

The second problem we study is that of selling multiple advertisement slots on a web page. This problem is more closely aligned with the problem faced by search engines. For each search keyword or web page, let the available advertisement slots be numbered in decreasing order of visibility. In the auctions currently being used by search engines, the advertisers interested in a particular keyword are first ranked according to some criterion (called the *ranking function*) and matched to the available slots in accordance with the ranking order obtained. The ranking function is, typically, an inherent part of the advertisement policy (or "philosophy") of the search engine. After pairing slots with advertisers, each advertiser who is assigned a slot is charged a price that is equal to the minimum bid required to retain her rank. This amount is, of course, no more than the bid of the advertiser. We show that

this pricing scheme is not truthful and gives an incentive for the advertisers to underbid, i.e. bid lower than their true valuation.

We develop a truthful auction for this problem under the assumption that the advertisers are not budget-constrained, i.e., they are willing to buy additional clicks as long as the marginal profit of getting an additional click is non-negative. This assumption is reasonable in a lot of scenarios, especially when the advertiser makes an immediate (expected) profit each time its advertisement is clicked on. Furthermore, since the ranking criterion (also referred to as the *ranking function*) is often an integral part of the advertisement policy of the search engine, we assume that the ranking function is specified as an input to the auction design problem. We consider the problem of designing auctions for a class of ranking functions which includes the current ranking functions being used by major search engines like Google and Yahoo!. For any ranking function in this class, we give an auction that ranks according to that function and show that it is the unique truthful auction that ranks according to the specified function. The uniqueness implies profit-maximality as a corollary. We also show that, under a reasonable assumption, our auction has a revenue equivalent to the auctions currently in use. These results are joint work with Ashish Goel and Rajeev Motwani, and are described in Chapter 6.

# Part I

# Privacy Protection

# Chapter 2

# K-Anonymity

There has been a tremendous growth in the amount of personal data that can be collected and analyzed. Data mining tools are increasingly being used to infer trends and patterns. In many scenarios, access to large amounts of *personal data* is essential in order for accurate inferences to be drawn. For example, at the beginning of an epidemic, a single hospital might see only a few isolated cases, whereas the combined patient pool of a group of hospitals might be sufficient to infer the outbreak of an epidemic. However, the use of data containing personal information has to be restricted in order to protect individual privacy. As discussed in the introduction, one solution is to release anonymized data that enables one to draw inferences about global trends without violating the privacy of individual records.

One approach to anonymization uses *perturbation* techniques in order to hide the exact values of the data [AS00, AA01, DN03, EGS03, DN04b, AST05, CDM+05]. However, this may not be suitable if one wants to draw inferences with 100% confidence. Another approach is to *suppress* some of the data values, while releasing the remaining data values exactly. We note that suppressing just the identifying attributes is not sufficient to protect privacy. For example, consider the following table which is part of a medical database, with the identifying attributes such as name and social security number removed.

| Age | Race | Gender | Zip Code | Diseases |
|---|---|---|---|---|
| 47 | White | Male | 21004 | Common Cold |
| 35 | White | Female | 21004 | Flu |
| 27 | Hispanic | Female | 92010 | Flu |
| 27 | White | Female | 92010 | Hypertension |

By joining this table with public databases (such as a voter list), non-identifying attributes, such as Age, Race, Gender and Zip Code in the above table, can together be used to identify individuals. In fact, Sweeney [Swe00] observed that for 87% of the population in the United States, the combination of Date of Birth, Gender and Zip Code corresponded to a unique person.

In order to ensure the protection of privacy, we adopt the $k$-Anonymity model that was proposed by Samarati and Sweeney [SS98, Sam01, Swe02]. Suppose we have a table consisting of $n$ tuples each having $m$ quasi-identifying attributes (Age, Race, Gender and Zip Code in the above table), and let $k > 1$ be an integer. The $k$-Anonymity framework provides for generalization of entries (generalization entails replacing an entry value with a less specific but semantically consistent value; a more formal description can be found in Section 2.1) in addition to suppression. The idea is to suppress/generalize some of the entries in the table so as to ensure that *for each tuple in the modified table, there are at least* $k - 1$ *other tuples in the modified table that are identical to it along the quasi-identifying attributes.* The objective is to minimize the extent of suppression and generalization. Note that entries in the column corresponding to the sensitive attribute ("Diseases" in the above example) are not altered. The following is an example of a $k$-anonymized table for $k = 2$.

| Age | Race | Gender | Zip Code | Diseases |
|---|---|---|---|---|
| * | White | * | 21004 | Common Cold |
| * | White | * | 21004 | Flu |
| 27 | * | Female | 92010 | Flu |
| 27 | * | Female | 92010 | Hypertension |

A $k$-anonymized table protects individual privacy in the sense that, even if an adversary has access to all the quasi-identifying attributes of all the individuals represented in the table, he would not be able to track down an individual's record further than a set of at least

$k$ records, in the worst case. Thus, $k$-anonymization of a table before its release prevents definitive *record linkages* with publicly available databases, and keeps each individual hidden in a crowd of $k-1$ other people. The privacy parameter $k$ must be chosen according to the application in order to ensure the required level of privacy.

## 2.1 Model and Results

We now formally define the problem of $k$-Anonymity and state our results. The input is a table having $n$ rows each with $m$ quasi-identifying attributes. We view the table as consisting of $n$ $m$-dimensional vectors: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Sigma^m$.

We first define a special case of the problem called *$k$-Anonymity with Suppression*, where suppression is the only permitted operation. A *$k$-Anonymous suppression function* $t$ maps each $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ by replacing some components of $\mathbf{x}_i$ by $*$ (which corresponds to hiding those components of $\mathbf{x}_i$), so that every $\tilde{\mathbf{x}}_i$ is identical to at least $k-1$ other $\tilde{\mathbf{x}}_j$s. This results in a partition of the $n$ row vectors into *clusters* of size at least $k$ each. The cost of the suppression, $c(t)$ is the total number of hidden entries, or equivalently, the total number of $*$s in all the $\tilde{\mathbf{x}}_i$s.

| $k$-Anonymity with Suppression: *Given $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \Sigma^m$, and an Anonymity parameter $k$, obtain a $k$-Anonymous suppression function $t$ so that $c(t)$ is minimized.* |
|---|

Next, we define the problem of *$k$-Anonymity with Generalization*, where in addition to suppressing entry values, we are also allowed to replace them with less specific but semantically consistent values. For example, we can make a date less specific by omitting the day and revealing just the month and year. We assume that for each attribute, a generalization hierarchy is specified as part of the input [SS98, Sam01]. For an attribute, each level of generalization corresponds to a partition of the attribute domain. A partition corresponding to any given level of the generalization hierarchy is a refinement of the partition corresponding to the next higher level. Singleton sets correspond to absence of generalization, while the partition consisting of a single set containing the whole domain corresponds to the highest level of generalization. Consider the example shown in Figure 2.1. The attribute "Quality" has a domain consisting of values $A+$, $A$, $A-$, $B+$, $B$ and

$B-$ and has two levels of generalization. In the absence of generalization, the value of this attribute is reported exactly. The first level of generalization corresponds to the partition $\{\{A+, A, A-\}, \{B+, B, B-\}\}$. In order to generalize an entry with value "A" to the first level of generalization, it is replaced with the set $\{A+, A, A-\}$. The next higher level of generalization (also the highest level in this case) corresponds to replacing the entry with the set containing the whole domain, which is equivalent to suppressing the entry.

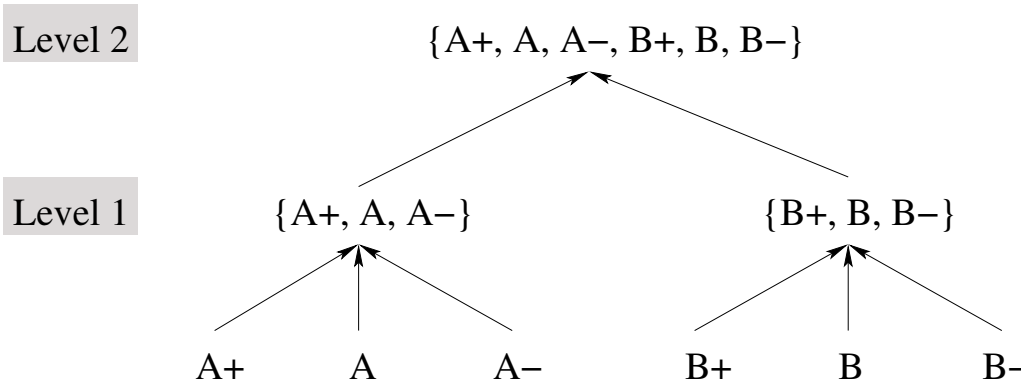



Figure 2.1: A possible generalization hierarchy for the attribute "Quality".

Let the $j^{th}$ attribute have domain $D^j$ and $l_j$ levels of generalization. Let the partition corresponding to the $h^{th}$ level of generalization be $D^j_h$ for $1 \leq h \leq l_j$, with $D^j_0 = D^j$. Let a value $y \in D^j$ when generalized to the $h^{th}$ level be denoted by $g_h(y)$, e.g., $g_1(A) = \{A+, A, A-\}$. A *generalization function* $h$ is a function that maps a pair $(i, j)$, $i \leq n$, $j \leq m$ to a level of generalization $h(i, j) \leq l_j$. Semantically, $h(i, j)$ denotes the level to which $j^{th}$ component of the $i^{th}$ vector (or the $(i, j)^{th}$ entry in the table) is generalized. Let $h(\mathbf{x}_i)$ denote the *generalized* vector corresponding to $\mathbf{x}_i$, i.e. $h(\mathbf{x}_i) = (g_{h(i,1)}(x_i[1]), g_{h(i,2)}(x_i[2]) \ldots, g_{h(i,m)}(x_i[m]))$. A generalization function is said to be $k$-Anonymous if for every $i$, $h(\mathbf{x}_i)$ is identical to $h(\mathbf{x}_j)$ for at least $k-1$ values of $j \neq i$.

Consider a $k$-Anonymous generalization function $h$. It incurs a cost of $r/l_j$ whenever it generalizes a value for the $j^{th}$ attribute to the $r^{th}$ level. The total cost incurred by the generalization function $h$ is defined as the sum of the costs incurred over all the entries of the table, i.e. cost$(h) = \sum_i \sum_j h(i, j)/l_j$. Now we are ready to give a formal definition of the problem.

$k$-Anonymity with Generalization: *Given $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \Sigma^m$, and an Anonymity parameter $k$, obtain a $k$-Anonymous generalization function $h$ such that cost$(h)$ is minimized.*

Note that the problem of $k$-Anonymity with Suppression is a special case of the problem of $k$-Anonymity with Generalization, with only one level of generalization (corresponding to hiding the entry completely) for every attribute.

Clearly the decision version of both of these problems is in NP, since we can verify in polynomial time if the solution is $k$-Anonymous and the suppression cost less than a given value. We show that $k$-Anonymity with Suppression is NP-hard even when the alphabet size $|\Sigma| = 3$. Note that this automatically implies NP-hardness of $k$-Anonymity with Generalization. This improves upon the NP-hardness result of [MW04] which required an alphabet size of $n$. On the positive side, we provide an $O(k)$-approximation algorithm for $k$-Anonymity with Generalization for arbitrary $k$ and arbitrary alphabet size. For a binary alphabet, we provide improved approximation algorithms for $k = 2$ (an approximation factor of $1.5$) and $k = 3$ (an approximation factor of 2).

The rest of the chapter is organized as follows. We establish the NP-hardness of $k$-Anonymity with Suppression in Section 2.2. We then present an $O(k)$-approximation algorithm for $k$-Anonymity with Generalization in Section 2.3. Next, in Sections 2.4 and 2.5, we provide a 1.5 approximation algorithm for the 2-Anonymity problem with binary alphabet, and a 2-approximation algorithm for 3-Anonymity with binary alphabet.

## 2.2 NP-hardness of $k$-Anonymity with Suppression

**Theorem 2.2.1** *$k$-Anonymity with Suppression is NP-hard even for a ternary alphabet, i.e., $(\Sigma = \{0, 1, 2\})$.*

**Proof:** In this proof, $k$-Anonymity refers to the problem of $k$-Anonymity with Suppression. We give a reduction from the NP-hard problem of EDGE PARTITION INTO TRIANGLES [Kan94] which is defined as follows: *Given a graph $G = (V, E)$ with $|E| = 3m$ for some integer $m$, can the edges of $G$ be partitioned into $m$ edge-disjoint triangles?*

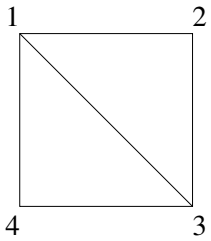
Given an instance of the above problem, $G = (V, E)$ with $3m$ edges (since the above problem is NP-hard even for simple graphs, we will assume that the graph G is simple), we create a preliminary table $T$ with $3m$ rows — one row for each edge. For each of the $n$ vertices of $G$, we create an attribute (column). The row corresponding to edge $(a, b)$, referred to as $r_{ab}$, has ones in the positions corresponding to $a$ and $b$ and zeros everywhere else. Let a star with four vertices (having one vertex of degree 3) be referred to as a 4-star.

**Equivalence to edge partition into triangles and 4-stars.** We first show that the cost of the optimal 3-Anonymity solution for the table $T$ is at most $9m$ if and only if $E$ can be partitioned into a collection of $m$ disjoint triangles and 4-stars. First suppose that such a partition of edges is given. Consider any triangle (with $a, b, c$ as its vertices). By suppressing the positions $a$, $b$ and $c$ in the rows $r_{ab}, r_{bc}$ and $r_{ca}$, we get a cluster containing three rows, with three $*$s in each modified row. Now consider a 4-star with vertices $a, b, c, d$, where $d$ is the center vertex. By suppressing the positions $a$, $b$ and $c$ in the rows $r_{ad}, r_{bd}$ and $r_{cd}$, we get a cluster containing three rows with three $*$s in each modified row. Thus we obtain a solution to 3-Anonymity of cost $9m$.

On the other hand, suppose that there is a 3-Anonymity solution of cost at most $9m$. Since $G$ is simple, any three rows are distinct and differ in at least 3 positions. Hence there should be at least three $*$s in each modified row, so that the cost of the solution is at least $9m$. This implies that the solution cost is exactly $9m$ and each modified row has exactly three $*$s. Since any cluster of size more than three will have at least four $*$s in each modified row, it follows that each cluster has exactly three rows. There are exactly two possibilities: the corresponding edges form either a triangle or a 4-star, and each modified row in a triangle has three $*$s and zeros elsewhere while each modified row in a 4-star has three $*$s, single 1 and zeros elsewhere. Thus, the solution corresponds to a partition of the edges of the graph into triangles and 4-stars.

**Equivalence to edge partition into triangles.** Since we want a reduction from EDGE PARTITION INTO TRIANGLES, we create a table $T'$ by "replicating" the columns of $T$ so as to force the 4-stars to pay more $*$s. Let $t = \lceil \log_2(3m + 1) \rceil$. In the new table $T'$, every row has $t$ *blocks*, each of which has $n$ columns. Consider an arbitrary ordering of the edges

in $E$ and express the rank of an edge $e = (a, b)$, in this ordering, in binary notation as $e_1 e_2 \ldots e_t$. In the row corresponding to edge $e$, each block has zeros in all positions except $a$ and $b$. A block can be in one of two configurations: $conf_0$ has a 1 in position $a$ and a 2 in position $b$ while $conf_1$ has a 2 in position $a$ and a 1 in position $b$. The $i^{th}$ block in the row corresponding to $e$ has configuration $conf_{e_i}$. For example, consider the graph shown in Figure 2.2. Suppose the edges $(3,4), (1,4), (1,2), (1,3), (2,3)$ are ranked 1 (i.e. $(001)_2$) through 5 (i.e. $(101)_2$) respectively. Then, the table in Figure 2.2 represents the 3-Anonymity instance corresponding to the graph, with the $i^{th}$ row in the table representing the vector corresponding to the edge ranked $i$.




| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (3,4) | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | 0 | 0 | 2 | 1 |
| (1,4) | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 2 |
| (1,2) | 1 | 2 | 0 | 0 | 2 | 1 | 0 | 0 | 2 | 1 | 0 | 0 |
| (1,3) | 2 | 0 | 1 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 2 | 0 |
| (2,3) | 0 | 2 | 1 | 0 | 0 | 1 | 2 | 0 | 0 | 2 | 1 | 0 |

Figure 2.2: The table shows the 3-anonymity instance corresponding to the graph on the left when the edges $(3,4), (1,4), (1,2), (1,3), (2,3)$ are ranked 1 through 5 respectively.

We will now show that the cost of the optimal 3-Anonymity solution on $T'$ is at most $9mt$ if and only if $E$ can be partitioned into $m$ disjoint triangles.

Suppose that $E$ can be partitioned into $m$ disjoint triangles. As earlier, every triangle in such a partition corresponds to a cluster with $3t$ $*$s in each modified row. Thus we get a 3-Anonymity solution of cost $9mt$.

For the converse, suppose that we are given a 3-Anonymity solution of cost at most $9mt$. Again, any three rows differ in at least $3t$ positions so that the cost of any solution is at least $9mt$. Hence the solution cost is exactly $9mt$ and each modified row has exactly $3t$ $*$s. Thus, each cluster has exactly three rows. We claim that the corresponding edges should form a triangle. We can see this as follows: suppose to the contrary the three rows form a 4-star. Let the common vertex be $v$. Consider the ternary digit $\in \{1, 2\}$ assigned by each of the three edges to $v$ in $conf_0$ — two of the three edges must have assigned the same digit to $v$. Since these two edges differ in rank, they must have a different configuration (and

therefore, a different digit in the column corresponding to $v$) in at least one of the blocks. Thus, the rows corresponding to the three edges contain an additional $*$ corresponding to vertex $v$ in addition to the $3t$ $*$s corresponding to the remaining three vertices, contradicting the fact that each row has exactly $3t$ $*$s. □

The above proof shows that k-Anonymity is NP-hard even with a ternary alphabet for $k = 3$. By reduction from EDGE PARTITION INTO $r$-CLIQUES [Kan94], we can extend the above proof for $k = \binom{r}{2}$, for $r \geq 3$. By replicating the graph in the above reduction, we can further extend the proof for $k = \alpha\binom{r}{2}$ for any integer $\alpha$ and $r \geq 3$.

## 2.3 Algorithm for General $k$-Anonymity

In this section, we study the problem of $k$-Anonymity with Generalization for general $k$ and arbitrary alphabet size, and give an $O(k)$-approximation algorithm for the problem. In this section, $k$-Anonymity refers to the problem of $k$-Anonymity with Generalization.

**Construction of Graph.** Given an instance of the $k$-Anonymity problem, we create an edge-weighted complete graph $G = (V, E)$. The vertex set $V$ contains a vertex corresponding to each vector in the $k$-Anonymity problem. For two rows $\mathbf{a}$ and $\mathbf{b}$, let the unscaled generalization cost for the $j^{th}$ component, $h_{a,b}(j)$, refer to the lowest level of generalization for attribute $j$ for which the $j^{th}$ components of both $\mathbf{a}$ and $\mathbf{b}$ are in the same partition, i.e. the lowest level for which both have the same generalized value. The weight, $w(e)$, of an edge $e = (a, b)$ is the sum over all components $j$ of the scaled generalization cost, i.e. $w(e) = \sum_j h_{a,b}(j)/l_j$ (recall that the scaling factor $l_j$ corresponds to the total number of levels of generalizations for the $j^{th}$ attribute). The $j^{th}$ attribute is said to contribute a weight of $h_{a,b}(j)/l_j$ to the edge $e$.

**Limitations of the Graph Representation.** As mentioned in the introduction, with this representation, we lose some information about the structure of the problem, and cannot achieve a better than $\Theta(k)$ approximation factor for the $k$-Anonymity problem. We show this by giving two instances (on binary alphabet) whose $k$-Anonymity cost differs by a factor of $\Theta(k)$, but the corresponding graphs for both the instances are identical. Let $l =$

$2^{k-2}$. For the first instance, take $k$ vectors with $kl$-dimensions each. The bit positions $(i-1)l+1$ to $il$ are referred to as the $i^{th}$ block of a vector. The $i^{th}$ vector has ones in the $i^{th}$ block and zeros everywhere else. The $k$-Anonymity cost for this instance is $k^2l$. For the second instance, take $k$ vectors with $4l = 2^k$ dimensions each. The $i^{th}$ vector breaks up its $2^k$ dimensions into $2^i$ equal-sized blocks and has ones in the odd blocks and zeros in the even blocks. This instance incurs a $k$-Anonymity cost of $4kl$. Note that the graph corresponding to both the instances is a $k$-clique with all the pairwise distances being $2l = 2^{k-1}$.

**Definition 2.1 (Charge of a vertex)** *For any given $k$-Anonymity solution, define the* charge *of a vertex to be the total generalization cost of the vector it represents.*

**Idea Behind the Algorithm.** Let $OPT$ denote the cost of an optimal $k$-Anonymity solution, i.e., $OPT$ is the sum of the charges of all the vertices in an optimal $k$-Anonymity solution. Let $F = \{T_1, T_2, \ldots, T_s\}$, a spanning forest (i.e. a forest containing all the vertices) in which each tree $T_i$ has at least $k$ vertices, be a subgraph of $G$. This forest describes a feasible partition for the $k$-Anonymity problem. In the $k$-Anonymity solution as per this partition, the charge of each vertex is no more than the weight of the tree containing the vertex; recall that the weight of a tree $T_i$ is given by $W(T_i) = \Sigma_{e \in E(T_i)} w(e)$, where $E(T_i)$ denotes the set of edges in tree $T_i$. We can see this as follows: if attribute $j$ has to be generalized to level $r$ for the vertices in tree $T_i$ (note that an attribute is generalized to the same level for all rows in a cluster), there must exist a pair of vertices $(a, b)$ in the cluster which have an unscaled generalization cost $h_{a,b}(j)$ equal to $r$. Thus, attribute $j$ contributes a weight of at least $r/l_j$ to the length of all paths (in G) between $a$ and $b$. In particular, attribute $j$ contributes a weight of at least $r/l_j$ to the weight of tree $T_i$. Next, we sum the charges of all the vertices to get that the $k$-Anonymity cost of the partition corresponding to the forest $F$ is at most $\Sigma_i |V(T_i)| W(T_i)$. We will refer to this as the $k$-Anonymity cost of the forest. Note that the weight of a forest is simply the sum of the weights of its trees. Hence, the ratio of the $k$-Anonymity cost to the weight of a forest is at most the number of vertices in the largest tree in the forest. This implies that if we can find a forest with the size of the largest component at most $L$ and weight at most $OPT$, then we have an

$L$-approximation algorithm. Next, we present an algorithm that finds such a forest with $L \leq \max\{2k-1, 3k-5\}$.

The algorithm has the following overall structure, which is explained in more detail in the next two subsections.

**Outline of the Algorithm:**

1. Create a forest $G$ with cost at most $OPT$. The number of vertices in each tree is at least $k$.

2. Compute a decomposition of this forest (deleting edges is allowed) such that each component has between $k$ and $\max\{2k-1, 3k-5\}$ vertices. The decomposition is done in a way that does not increase the sum of the costs of the edges.

### 2.3.1 Algorithm for Producing a Forest with Trees of Size at least $k$

The key observation is that since each partition in a $k$-Anonymity solution groups a vertex with at least $k-1$ other vertices, the charge of a vertex is at least equal to its distance to its $(k-1)^{st}$ nearest neighbor. The idea is to construct a directed forest such that each vertex has at most one outgoing edge and $(\overrightarrow{u,v})$ is an edge only if $v$ is one of the $k-1$ nearest neighbors of $u$.

**Algorithm** FOREST
**Invariant:**

- The chosen edges do not create any cycle.
- The out-degree of each vertex is at most one.

1. Start with an empty edge set so that each vertex is in its own connected component.

2. Repeat until all components are of size at least $k$:

   Pick any component $T$ having size smaller than $k$. Let $u$ be a vertex in $T$ without any outgoing edges. Since there are at most $k-2$ other vertices in $T$, one of the $k-1$ nearest neighbors of $u$, say $v$, must lie outside $T$. We add the edge $(\overrightarrow{u,v})$ to the forest. *Observe that this step does not violate any of the invariants.*

**Lemma 2.3.1** *The forest produced by algorithm* FOREST *has minimum tree size at least $k$ and has cost at most $OPT$.*

**Proof:** It is evident from the algorithm description that each component of the forest it produces has at least $k$ vertices.

Let the cost of an edge $(\overrightarrow{u,v})$ be paid by vertex $u$. Note that each vertex $u$ pays for at most one edge to one of its $k-1$ nearest neighbors. As noted earlier, this is less than the charge of this vertex in any $k$-Anonymity solution. Thus, the sum of costs of all edges in the forest is less than $OPT$, the total charge of all vertices in an optimal solution. □

### 2.3.2 Algorithm to Decompose Large Components into Smaller Ones

We next show how to break any component with size greater than $\max\{2k-1, 3k-5\}$ into two components each of size at least $k$. Let the size of the component we are breaking be $s > \max\{2k-1, 3k-5\}$.

**Algorithm** DECOMPOSE-COMPONENT

1. Pick any vertex $u$ as the candidate vertex.
2. Root the tree at the candidate vertex $u$. Let U be the set of subtrees rooted at the children of $u$. Let the size of the largest subtree of $u$ be $\phi$, rooted at vertex $v$. If $s-\phi \geq k-1$, then we do one of the following partition and terminate (see Figure 2.3).
   A. If $\phi \geq k$ and $s-\phi \geq k$, then partition the tree into the largest subtree and the rest.
   B. If $s-\phi = k-1$, partition the tree into a component containing the subtrees rooted at the children of $v$ and the rest. To connect the children of $v$ create a dummy vertex $v'$ to replace $v$. Note that $v'$ is only a Steiner vertex (see Figure 2.4) and does not contribute to the size of the first component. Clearly, the sizes of both the components are at least $k$.
   C. If $\phi = k-1$, then partition into a component containing the subtree rooted at $v$ along with the vertex $u$ and the rest. In order to connect the children of $u$ in the second component, we create a Steiner vertex $u'$.

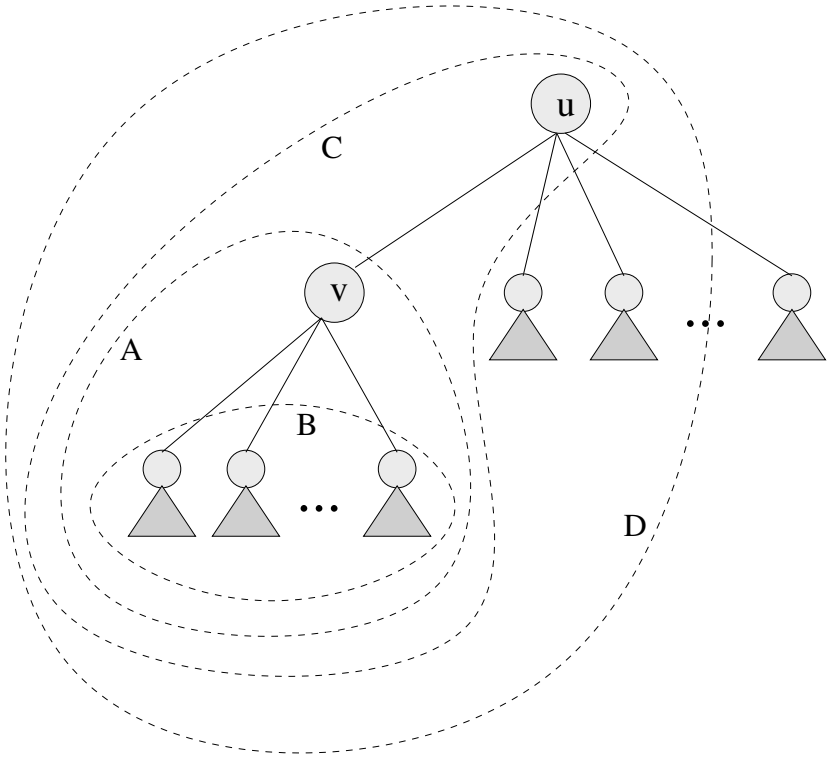



Figure 2.3: The decompositions corresponding to the sub-cases of the algorithm DECOMPOSE-COMPONENT.

D. Otherwise, all subtrees have size at most $k-2$. In this case, we create an empty partition and keep adding subtrees of $u$ to it until the first time its size becomes at least $k-1$. Clearly, at this point, its size is at most $2k-4$. Put the remaining subtrees (containing at least $k-1$ vertices, since there are at least $3k-4$ vertices in all) into the other partition. Observe that since $s \geq 2k$, at most one of the partitions has size equal to $k-1$. If such a partition exists, add $u$ to that partition, else add $u$ to the first partition. In order to keep the partition not containing $u$ connected, a Steiner vertex $u'$ corresponding to $u$ is placed in it.

3. Otherwise, pick the root of the largest subtree $v$ as the new candidate vertex and go to Step 2.

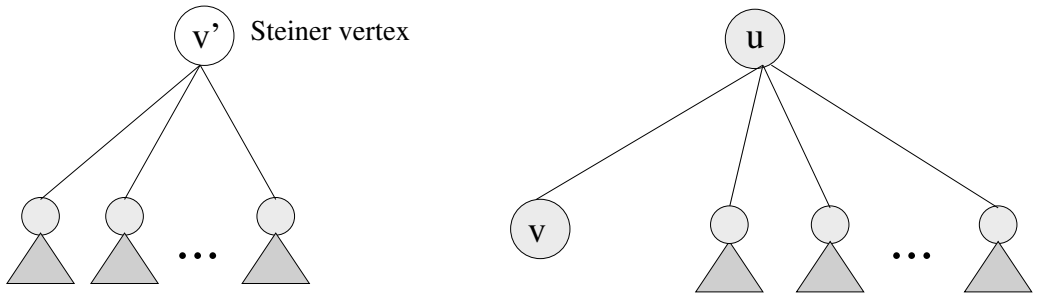



Figure 2.4: The decomposition corresponding to case B; the left partition contains a Steiner vertex $v'$ that does not contribute to its size.

**Lemma 2.3.2** *The above algorithm terminates.*

**Proof:** We will prove this by showing that the size of the largest component $\phi$ (in Step 2) decreases in each iteration. Consider moving from candidate vertex $u$ in one iteration to candidate vertex $v$ in the next iteration. Since the algorithm did not terminate with $u$, if we root the tree at $v$, then the size of the subtree rooted at $u$ is less than $k-1$. When we consider the largest subtree under $v$, either it is rooted at $u$, in which case, it is smaller than $k-1 < s-(k-1)$ and the algorithm terminates in this step; otherwise, the new largest subtree is a subtree of the previous largest subtree. □

**Theorem 2.3.3** *There is a polynomial-time algorithm for the $k$-Anonymity problem, that achieves an approximation ratio of* $\max\{2k-1, 3k-5\}$.

**Proof:** First, use Algorithm FOREST to create a forest with cost at most $OPT$ and minimum tree size at least $k$. Then repeatedly apply Algorithm DECOMPOSE-COMPONENT to any component that has size larger than $\max\{2k-1, 3k-5\}$. Note that both these algorithms terminate in $O(kn^2)$ time. □

The above algorithm can also be used when the attributes are assigned weights and the goal is to minimize the weighted generalization cost. In this case, the cost contributed by an attribute to an edge in the graph $G$ is multiplied by its weight. The rest of the algorithm proceeds as before. It is also easy to extend the above analysis to the version of the problem where we allow an entire row to be deleted from the published database, instead of forcing it to pair with at least $k-1$ other rows. The deletion of an entire row is

modeled as suppressing all the entries of that row (or generalizing all the entries of that row to the highest level). The objective function is the same as before: minimize the overall generalization cost. We first note that the distance between any two vertices is no more than the cost of deleting a vertex. Thus, if we run the same algorithm as above, the total cost of the forest $F$ produced by Algorithm FOREST is no more than the optimal $k$-Anonymity cost (this is because the charge of any vertex in the optimal $k$-Anonymity solution is still no less than its distance to its $(k-1)^{st}$ nearest neighbor). The analysis for the rest of the algorithm remains the same.

## 2.4 Improved Algorithm for $2$-Anonymity

In this section, we study the special case of $k = 2$. The algorithm of the previous section gives a 3-approximation algorithm for this case. We improve upon this result for binary alphabet, and provide a polynomial-time $1.5$-approximation algorithm for $2$-Anonymity (note that for binary alphabet, generalization is equivalent to suppression). This algorithm uses a technique that is completely different from the previous algorithm, and could potentially be extended to get an improved approximation factor for the general case. For this algorithm, we use the minimum-weight $[1, 2]$-factor of a graph constructed from the $2$-Anonymity instance. A $[1, 2]$-factor of an edge-weighted graph $G$ is defined to be a spanning (i.e., containing all the vertices) subgraph $F$ of $G$ such that each vertex in $F$ has degree $1$ or $2$. The weight of $F$ is the sum of the weights of the edges in $F$. Cornuejols [Cor88] showed that a minimum-weight $[1, 2]$-factor of a graph can be computed in polynomial time.

Given an instance of the 2-Anonymity problem on binary alphabet, we create an edge-weighted complete graph $G = (V, E)$ as follows. The vertex set $V$ contains a vertex corresponding to each vector in the 2-Anonymity problem. The weight of an edge $(a, b)$ is the Hamming distance between the vectors represented by $a$ and $b$ (i.e., the number of positions at which they differ). First we obtain a minimum-weight $[1, 2]$-factor $F$ of $G$. By optimality, $F$ is a vertex-disjoint collection of edges and pairs of adjacent edges (if a $[1, 2]$-factor has a component which is either a cycle or a path of length $\geq 3$, we can obtain a $[1, 2]$-factor of smaller weight by removing edge(s)). We treat each component of $F$ as a

*cluster*, i.e., retain the bits on which all the vectors in the cluster agree and replace all other bits by $*$s. Clearly, this results in a 2-anonymized table.

**Theorem 2.4.1** *The number of $*$s introduced by the above algorithm is at most $1.5$ times the number of $*$s in an optimal 2-Anonymity solution.*

Before we prove this theorem, consider three $m$-bit vectors $x_1, x_2$ and $x_3$ with pairwise Hamming distances $\alpha, \beta$ and $\gamma$ as shown in Figure 2.5. Without loss of generality, let $\gamma \geq \alpha, \beta$. Let $x_{med}$ denote the *median* vector whose $i^{th}$ bit is the majority of the $i^{th}$ bits of $x_1, x_2$ and $x_3$ and let $p, q$ and $r$ be the Hamming distances to $x_{med}$ from $x_1$, $x_2$ and $x_3$ respectively. Let $x_s$ be the *star* vector obtained by minimal suppression of $x_1, x_2$ and $x_3$, i.e., it has the common bits where the three vectors agree and $*$s elsewhere. Observe that $\alpha = q + r$, $\beta = r + p$ and $\gamma = p + q$. The other relevant distances are shown in the figure.

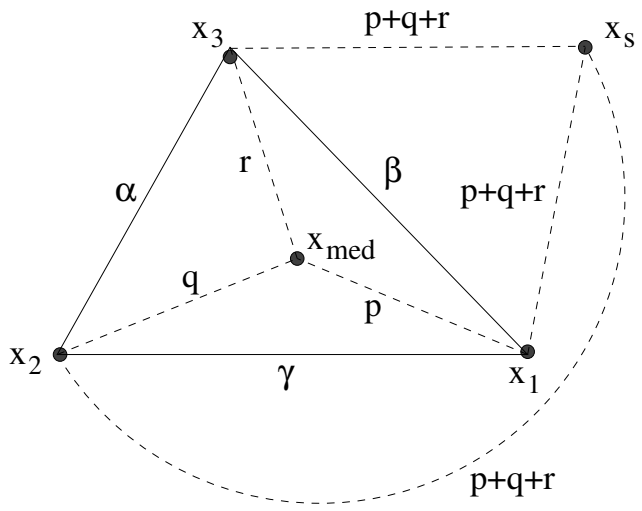



Figure 2.5: Three vectors and their corresponding "median" and "star" vectors

**Observation 2.4.2** *If vertices $x_1$, $x_2$ and $x_3$ (as shown in Figure 2.5) form a cluster in a $k$-Anonymity solution, the number of $*$s in each modified vector is exactly equal to $p+q+r = \frac{1}{2}(\alpha + \beta + \gamma)$. If the cluster contains additional vertices, then the number of $*$s is at least $\frac{1}{2}(\alpha + \beta + \gamma)$.*

To see this, first note that since $x_{med}$ is the median vertex, the attributes that contribute to $p$, $q$ and $r$ are distinct. Therefore, the number of $*$s in each modified vector is at least $p + q + r$. Moreover, when $x_1$, $x_2$ and $x_3$ are the only three vertices in the cluster, each attribute corresponding to a $*$ in the modified vector contributes to exactly one of $p$, $q$ and $r$.

Let $c_{OFAC}$ denote the weight of an optimal $[1, 2]$-factor, let $c_{ALG}$ be the cost of the 2-Anonymity solution obtained from it and let $OPT$ denote the cost of the optimal 2-Anonymity solution respectively. The optimal 2-Anonymity solution can be assumed to consist only of disjoint clusters of size $2$ or $3$ (as bigger clusters can be broken into such clusters without increasing the cost). We can derive a $[1, 2]$-factor from this solution as follows: for each cluster of size 2, include the edge between the two vertices; for a cluster of size 3, include the two lighter edges of the triangle formed by the three vertices. Denote the weight of this $[1, 2]$-factor by $c_{FAC}$.

**Lemma 2.4.3** $c_{ALG} \leq 3 \cdot c_{OFAC}$

**Proof:** Consider the optimal $[1, 2]$-factor and the $k$-Anonymity solution corresponding to it. For a cluster of size 2, we have to suppress all the bits at which the two vectors differ so that the total number of $*$s in the two rows is twice the Hamming distance (which is equal to the edge weight). For a cluster of size 3, say the one in the figure, by Observation 2.4.2, the number of $*$s in each row is exactly $(\alpha + \beta + \gamma)/2$. So, the total number of stars is $\frac{3}{2}(\alpha + \beta + \gamma) \leq 3(\alpha + \beta)$ (using triangle inequality). The optimal $[1, 2]$-factor would have contained the two lighter edges of the triangle, incurring a cost of $(\alpha + \beta)$ for this cluster. Summing over all the clusters formed by the optimal $[1, 2]$-factor algorithm, we get $c_{ALG} \leq 3 \cdot c_{OFAC}$. □

**Lemma 2.4.4** $c_{FAC} \leq \frac{1}{2}OPT$

**Proof:** Consider the optimal $k$-Anonymity solution and the $[1, 2]$-factor corresponding to it. For a cluster of size 2, cost incurred by the $[1, 2]$-factor $FAC$ is equal to half the cost incurred in $OPT$. For a cluster of size 3, say the one in Figure 2.5, cost incurred in $FAC$ is equal to $\alpha + \beta \leq \frac{2}{3}(\alpha + \beta + \gamma) = \frac{4}{3}(p + q + r)$, where the inequality is obtained by using the fact $\gamma \geq \alpha, \beta$. Since the cost incurred in $OPT$ is $3(p + q + r)$, cost incurred in

$FAC$ is at most half the cost incurred in $OPT$. By summing over all the clusters, we get $c_{FAC} \leq OPT/2$. □

Since $c_{OFAC} \leq c_{FAC}$, it follows from the above lemmas that $c_{ALG} \leq \frac{3}{2}OPT$, which proves Theorem 2.4.1. For an arbitrary alphabet size, $x_{med}$ is no longer defined. However, it can be shown that $OPT \geq (\alpha + \beta + \gamma) \geq \frac{3}{2}(\alpha + \beta)$, proving $c_{FAC} \leq \frac{2}{3}OPT$. Since $c_{ALG} \leq 3 \cdot c_{OFAC}$ holds as before, we get $c_{ALG} \leq 2 \cdot OPT$. Thus, the same algorithm achieves a factor 2 approximation for 2-Anonymity with Suppression for arbitrary alphabet size.

## 2.5 Improved Algorithm for 3-Anonymity

We now present a 2-approximation algorithm for 3-Anonymity with a binary alphabet (again generalization is equivalent to suppression in this case). The idea is similar to the algorithm for 2-Anonymity. We construct the graph $G$ corresponding to the 3-Anonymity instance as in the previous algorithm. A 2-factor of a graph is a spanning subgraph with each vertex having degree 2 (in other words, a collection of vertex-disjoint cycles spanning all the vertices). We first run the polynomial-time algorithm to find a minimum-weight 2-factor $F$ of the graph $G$ [Cor88]. We show that the cost of this 2-factor, say $c_{OFAC}$, is at most $2/3$ times the cost of the optimal 3-Anonymity solution, say $OPT$. Then, we show how to transform this 2-factor $F$ into a 3-Anonymity solution $ALG$ of cost $c_{ALG} \leq 3 \cdot c_{OFAC}$, giving us a factor-2 approximation algorithm for 3-Anonymity.

**Lemma 2.5.1** *The cost of the optimal 2-factor, $c_{OFAC}$ on graph $G$ corresponding to the vectors in the 3-Anonymity instance is at most $\frac{2}{3}$ times the cost of the optimal 3-Anonymity solution, $OPT$.*

**Proof:** Consider the optimal 3-Anonymity solution. Observe that it will cluster 3, 4 or 5 vertices together (any larger groups can be broken up into smaller groups of size at least 3, without increasing the cost of the solution). Given an optimal solution to the 3-Anonymity problem, we construct a 2-factor solution as follows: for every cluster of the 3-Anonymity solution, pick the minimum-weight cycle involving the vertices of the cluster. Next, we analyze the cost $c_{FAC}$ of this 2-factor. Define the *charge* of a vertex to be the number of

$*$s in the vector corresponding to this vertex in the 3-Anonymity solution. We consider the following three cases:

(a) If a cluster $i$ is of size 3, the 2-factor contains a triangle on the corresponding vertices. Let $a$, $b$ and $c$ be the lengths of the edges of the triangle. By Observation 2.4.2, we get that $(a+b+c)$ is twice the charge of each vertex in this cluster. Thus, $OPT$ pays a total cost of $OPT_i = \frac{3}{2}(a+b+c)$ while $FAC$ pays $c_{FAC,i} = a+b+c = \frac{2}{3}OPT_i$.

(b) If a cluster $i$ is of size 4, the 2-factor corresponds to the cheapest 4-cycle on the four vertices. Let $\tau$ be the sum of the weights of all the $\binom{4}{2} = 6$ edges on these four vertices. Consider the three 4-cycles on these vertices. As each edge appears in two 4-cycles, the average cost of a 4-cycle is $\frac{2}{3}\tau$. By choosing the minimum weight 4-cycle, we ensure that the cost paid by $FAC$ for these vertices $c_{FAC,i} \leq \frac{2}{3}\tau$. Also, by Observation 2.4.2, the charge of any of these 4 vertices is at least half the cost of any triangle on (three of) these four vertices. The cost of the most expensive triangle is at least equal to the average cost over all the $\binom{4}{3} = 4$ triangles, which is equal to $\frac{2}{4}\tau$ (since each edge appears in two triangles). Hence the cost paid by $OPT$, $OPT_i \geq 4 \cdot \frac{1}{2} \cdot \frac{2}{4} \cdot \tau = \tau$. Thus, $c_{FAC,i} \leq \frac{2}{3}OPT_i$.

(c) If a cluster $i$ is of size 5, let $\tau$ be the sum of weights of all $\binom{5}{2} = 10$ edges on these five vertices. By an argument similar argument to (b), $FAC$ pays $c_{FAC,i} \leq \frac{5}{10}\tau$. Also, the charge of any of these vertices is at least half the cost of any triangle on (three of) these vertices. Since the average cost of a triangle is $\frac{3}{10}\tau$, the number of $*$s in each vertex is at least $\frac{1}{2}\frac{3}{10}\tau$. Thus, cost paid by $OPT$ for cluster $i$, $OPT_i \geq 5 \cdot \frac{1}{2} \cdot \frac{3}{10} \cdot \tau = \frac{3}{4}\tau$. Thus, $c_{FAC,i} \leq \frac{2}{3}OPT_i$.

Thus, adding up over all clusters, we get $c_{FAC} \leq \frac{2}{3}OPT$. Thus, $c_{OFAC} \leq \frac{2}{3}OPT$. □

**Lemma 2.5.2** *Given a 2-factor $F$ with cost $c_F$, we can get a solution for 3-Anonymity of cost $c_{ALG} \leq 3 \cdot c_F$.*

**Proof:** To get a solution for 3-Anonymity, we make every cycle in $F$ with size 3, 4 or 5 into a cluster. Let $\mathrm{len}(C)$ denote the length of a cycle $C$ in the 2-factor. For each cycle larger $C$, if $\mathrm{len}(C) = 3x$ for $x$ an integer, then we decompose it into $x$ clusters, each containing

3 adjacent vertices of $C$. Similarly, if $\text{len}(C) = 3x + 1$, $x$ an integer, we decompose it into $x$ clusters: $x - 1$ of size 3, and one of size 4. If $\text{len}(C) = 3x + 2$, $x$ an integer, then we decompose it into $x - 2$ clusters of size 3, and two clusters of size $4$. In all these cases, of all the possible decompositions, we choose the one in which the total cost of edges of the cycle within the clusters is minimized. Depending on the size of the cycle $C$ in the 2-factor, we can show that the 3-Anonymity solution $ALG$ pays as follows:

(a) For a triangle, $ALG$ pays $3 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

(b) For a 4-cycle, $ALG$ pays at most $4 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

(c) For a 5-cycle, $ALG$ pays at most $5 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

The above inequalities follow from an observation similar to Observation 2.4.2, namely that the vertices of a cycle $C$ can differ in at most $\frac{1}{2}\text{len}(C)$ attributes.

(e) For a $(3x + 1)$-cycle, $x > 1$, $ALG$ pays at most $\frac{6(x-1)+12}{3x+1} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$. This is obtained by considering the minimum 3-Anonymity cost over the $(3x+1)$ possible decompositions into clusters. Each edge $e$ of the cycle $C$ appears in a cluster of size 4 in three decompositions and contributes a cost of at most $4w(e)$ to the $k$-Anonymity cost of the decomposition. In addition, each edge appears in a cluster of size 3 in $(2(x - 1))$ decompositions contributing a cost of at most $3w(e)$ to the $k$-Anonymity cost of these decompositions. Summing over all edges, the total $k$-Anonymity cost of all the $3x + 1$ decompositions is at most $(3 \cdot 2(x - 1) + 4 \cdot 3) \cdot \text{len}(C)$ and $ALG$ pays no more than the average cost of a decomposition.

(f) For a $(3x + 2)$-cycle, $x > 1$, $ALG$ pays at most $\frac{6(x-2)+24}{3x+2} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$. This is obtained by an analysis similar to (e) above. Note that we get a better bound on the cost by splitting into $x - 2$ clusters of size 3 and two clusters of size $4$, instead of $x - 1$ clusters of size 3 and one clusters of size $5$.

Thus, summing over all clusters, $ALG$ pays no more than three times the total cost of all cycles, i.e., $c_{ALG} \leq 3 \cdot c_F$. □

Note that the above analysis is tight, since equality can hold in case (f), when $x = 2$, e.g. for vectors $\{0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000\}$, where the optimal 2-factor is a cycle through all the vertices in the given order.

Combining the above lemmas, we obtain a factor 2 approximation for 3-Anonymity.

# Chapter 3

# Secure Computation of the $k^{th}$-ranked Element

In this chapter, we consider the problem of computing statistical functions over the union of large, confidential datasets held by different parties. In particular, we are interested in the problem of computing the $k^{th}$-*ranked element* of an ordered set $S$ split between two or more parties in a privacy-preserving manner.

For an ordered set $S \subset \mathbb{R}$, the $k^{th}$-ranked element is the value $x \in S$ that is ranked $k$ when the set $S$ is sorted in increasing order. Of particular interest is the median, which is the element with rank $p = \lceil |S|/2 \rceil$. Given two parties $A$ and $B$ with datasets $D_A, D_B \subset \mathbb{F}$, respectively, we consider the problem of privately computing the $k^{th}$-ranked element of $D_A \cup D_B$. We also consider this problem in the multi-party case.

We are interested in scenarios where the datasets $D_A$ and $D_B$ contain proprietary information, and neither party is willing to share its data with the other. Furthermore, the datasets involved are very large. For example, consider two health insurance companies wishing to compute the median life expectancy of the smokers insured by them. In such a setting, both the number of insured smokers as well as their life expectancies are private information, but the median life expectancy is of combined mutual interest. Another example is the annual Taulbee survey which collects salary and demographic data for faculty in computer science and computer engineering departments in North America. Typically, academic departments report only a small number of statistics like the minimum, maximum,

average and median salary for assistant, associate and full professor positions. The Taulbee survey is thus able to publish only limited aggregate information. A privacy-preserving, multi-party solution for the $k^{th}$-ranked element would enable universities to quickly compute the combined salary statistics without trusting individual salaries to Taulbee. Such a protocol would also facilitate the computation of histograms [GMP97, PGI99, JKM+98] in a privacy-preserving manner.

## Prior Work

The problem we discuss is referred to as *Secure Function Evaluation (SFE)* in the cryptography literature. It involves several parties with private inputs that wish to compute a function of their joint inputs, and require that the process of computing the function does not reveal to an adversarial party (or a coalition of such parties) any information that cannot be computed using the input of the adversary and the output of the function.

There exist well-known solutions that enable two or more parties to perform secure computation of any function [Yao86, GMW87]. The general method employed by these solutions is to construct a combinatorial circuit that computes the required function, and then run a distributed protocol that securely evaluates the circuit. We provide a brief sketch of a two-party protocol due to Yao in Section 3.2.2[1]. The communication overhead of these generic protocols is linear in the size of the circuit. The computation involves (at the least) running an oblivious transfer protocol for every input gate, or for every gate of the circuit, depending on the implementation. Let $M$ be the size of the domain $\mathbb{F}$ from which the datasets $D_A$ and $D_B$ are drawn, and let $n = |D_A| + |D_B|$ be the total number of the input elements. Then, the $k^{th}$-ranked element can be computed via a circuit of size $\Omega(n \log M)$ (since reading in the input requires at least $n \log M$ gates), which implies that for large values of $n$, the overhead of a secure protocol obtained from generic constructions is too large.

In another generic construction, Naor and Nissim [NN01] show that any two-party communication protocol can be translated into a secure computation protocol. Effectively,

[1] The interested reader can find a detailed description of these protocols in the references above. Alternatively, descriptions of the two-party protocols are available at, e.g., [LP02, Gol98], and descriptions of the multi-party protocols can be found, for example, in [BMR90, FY92, Gol98].

a protocol with communication complexity of $c$ bits is transformed into a secure protocol which performs $c$ invocations of oblivious transfer (or SPIR) from a database of length $2^c$. Since there is a protocol, due to Karchmer, for computing the median with $\log n$ communication [KN97], the implication is that the size of the database for the OT/SPIR invocations is polynomial in $n$, and the communication is $\log n$ times that of the OT/SPIR protocol. If the protocol uses SPIR based on the PIR protocol of Cachin et al. [CMS99], it obtains polylogarithmic communication overhead. The drawback, in addition to hidden constants, is that it requires application of a public-key operation to each item in the database, which implies that the number of public-key operations is polynomial in $n$.

## Contributions

The results in [Yao86, GMW87] and [NN01] are quite powerful in that they enable general transformations from known algorithms to secure protocols. Our interest, however, is to determine how efficiently a specific function, namely the $k^{th}$-ranked element, can be computed. We are motivated by applications where the total number of data points $n$ owned by the parties is very large, and thus even a linear communication and computation overhead might be prohibitive; even taking recent results on extending oblivious transfers [IKNP03] into account, the overhead is $\Omega(n)$. We describe protocols with sub-linear communication and computation overhead. Specifically, in the two-party case, we reduce the computation of the $k^{th}$-ranked element to $O(\log k)$ secure comparisons[2] of ($\log M$)-bit inputs, where $\log M$ is the number of bits needed to describe the elements in the sets $D_A, D_B$. We also show how to obtain security against malicious adversaries. In the multi-party case, we reduce the computation of the $k^{th}$-ranked element to $O(\log M)$ simple secure computations that involve additions and a comparison of ($\log M$)-bit long numbers. Again, this protocol can be made secure against malicious adversaries. Interestingly, the multi-party solution can be applied to the two-party scenario if it uses secure two-party protocols as primitives. This is in contrast to the typical case in secure computation where secure multi-party protocols require the presence of an honest majority, which is not available in the two-party case. While the two-party protocol requires inputs comprising of distinct values, the multi-party

[2] If the two parties possess inputs $x$ and $y$, a *secure comparison* reveals 0 if $x \geq y$ and 1 otherwise, and nothing more.

protocol can be applied directly even to inputs that contain duplicate items. The advantage of our two-party solution is that the number of rounds is logarithmic in the number of input items, whereas the number of rounds of the multi-party solution is logarithmic in the size of the domain of possible input values.

The protocols given here are modifications of well known algorithms in the communication complexity literature [Rod82, KN97]. Our contribution is the modifications and proofs of security that result in privacy-preserving solutions, for both semi-honest and malicious adversaries. In addition, we show how the parties can compute the $k^{th}$-ranked element while hiding the actual sizes of their databases from each other. We note that the communication complexity lower bound for computing the median is $\min\{\log n, \log M\}$ [KN97] whereas our result entails a communication cost of $O(\log n \cdot \log M)$ for a *secure* computation.

## Efficient Secure Computation via Reduction and Composition

We take the same approach as that of previous solutions for secure computation over large inputs (e.g. [LP02, FIM+01, CIK+01]), and reduce this task to many invocations of secure computation of simpler functions of small inputs (but unlike these constructions, we also design protocols which are secure against malicious adversaries). That is, we describe a protocol for computing the $k^{th}$-ranked element that uses oracle queries to a few simple functionalities and is secure if these functionalities are computed by a trusted oracle. A composition theorem (see [Can00, Can01] and discussions below) shows that if the oracle queries are replaced by secure protocols, then the resulting combined protocol is also secure. In the semi-honest case, the oracle queries can be replaced by very simple invocations of secure function evaluation. In the malicious adversary case, they are replaced by a reactive secure computation of a simple function. The result of the reduction is a distributed protocol whose overhead is sub-linear in the size of the inputs and is actually feasible even for very large inputs. We also note that the protocol computes the *exact* value of the $k^{th}$-ranked item, rather than computing an approximation.

## 3.1 Security Definitions and a Composition Theorem

We describe protocols that are secure against malicious adversaries; we therefore use definitions that compare the actual execution of the protocol to an *ideal* implementation, rather than use definitions that use simulation. The definitions we use follow those of Canetti [Can00] and Goldreich [Gol98]. We also state a composition theorem that is used in proving the security of the protocols.

A *semi-honest adversary* is an adversary that follows the instructions of the protocol. It might try, however, to use the information that it learns during the execution to learn information about the inputs of the other parties. A *malicious adversary* is an adversary that can behave arbitrarily. In particular, there are several things that a malicious adversary can do which we cannot hope to avoid: (1) it can refuse to participate in the protocol, (2) it can substitute an arbitrary value for its input, and (3) it can abort the protocol prematurely. Following [Can00, Gol98] we do not consider solutions to the early termination problem (i.e. item (3) above), also known as the fairness issue, since there is no perfect solution for this issue and existing solutions are quite complex. Furthermore, premature termination of the protocol by one party is detected by the other parties which, in many scenarios, can then take measures against the corrupt party. This is different than other types of malicious activity which are not easily detected.

The security definition we use captures both the correctness and the privacy of the protocol. We only provide definitions for the two-party case. For a detailed discussion of security definitions, for the two-party and multi-party scenarios, we refer the reader to [Can00, Gol98]. The definition is based on a comparison to the ideal model in which there is a trusted third party (TTP), and the corrupt parties can choose to give any arbitrary input to the trusted party, and to terminate the protocol prematurely, even at a stage where they have received their output and the other parties have not. We limit it to the case where both parties compute the same function $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$.

**Definition 3.1 (The Ideal Model)** *A strategy for party $A$ in the ideal model is a pair of PPT (probabilistic polynomial time) algorithms, $A_I(X, r)$ that uses the input $X$ and a sequence of coin flips $r$ to generate an input that $A$ sends to the trusted party, and $A_O(X, r, Z)$ which takes as an additional input the value $Z$ that $A$ receives from the TTP, and outputs*

*$A$'s final output. If $A$ is honest then $A_I(X, r) = X$ and $A_O(X, r, Z) = Z$. A strategy for party $B$ is similarly defined using functions $B_I(Y, r)$ and $B_O(Y, r, Z)$.*

*The definition is limited to the case where at least one of the parties is honest. We call an adversary that corrupts only one of the parties an* admissible adversary. *The joint execution of $A$ and $B$ in the ideal model, denoted by $\mathsf{IDEAL}_{A,B}(X, Y)$, is defined to be*

- *If $B$ is honest,*
  - $\mathsf{IDEAL}_{A,B}(X, Y)$ *equals* $(A_O(X, r, f(X', Y)), f(X', Y))$*, where* $X' = A_I(X, r)$ *(in the case that A did not abort the protocol),*
  - *or,* $\mathsf{IDEAL}_{A,B}(X, Y)$ *equals* $(A_O(X, r, f(X', Y)), -)$*, where* $X' = A_I(X, r)$ *(if A terminated the protocol prematurely).*
- *If $A$ is honest*
  - $\mathsf{IDEAL}_{A,B}(X, Y)$ *equals* $(f(X, Y'), B_O(Y, r, f(X, Y')))$*, where* $Y' = B_I(Y, r)$*,*
  - *or,* $\mathsf{IDEAL}_{A,B}(X, Y)$ *equals* $(-, B_O(Y, r, f(X, Y')))$*, where* $Y' = B_I(Y, r)$*.*

In the real execution, a malicious party could follow any strategy that can be implemented by a PPT algorithm. The strategy is an algorithm mapping a partial execution history to the next message sent by the party in the protocol.

**Definition 3.2 (The Real Model (for semi-honest and malicious adversaries))** *Let $f$ be as in Definition 3.1, and $\Pi$ be a two-party protocol for computing $f$. Let $(A', B')$ be a pair of PPT algorithms representing the parties' strategies. This pair is admissible w.r.t. $\Pi$ if at least one of $(A', B')$ is the strategy specified by $\Pi$ for the corresponding party. In the* semi-honest case*, the other party could have an arbitrary output function. In the* malicious *case, the other party can behave arbitrarily throughout that protocol.*

*The joint execution of $\Pi$ in the real model, denoted $\mathsf{REAL}_{\Pi,A',B'}(X, Y)$ is defined as the output pair resulting from the interaction between $A'(X)$ and $B'(Y)$.*

The definition of security states that an execution of a secure real model protocol under any admissible adversary can be simulated by an admissible adversary in the ideal model.

**Definition 3.3 (Security (for both the semi-honest case and the malicious case))** *Let $f$ and $\Pi$ be as in Definition 3.2. Protocol $\Pi$* **securely computes** *$f$ if for every PPT pair $(A', B')$ that is admissible in the real model (of Definition 3.2) there is a PPT pair $(A, B)$ that is admissible in the ideal model (of Definition 3.1), such that $\mathsf{REAL}_{\Pi,A',B'}(X, Y)$ is computationally indistinguishable from $\mathsf{IDEAL}_{A,B}(X, Y)$.*

## 3.2 Cryptographic Tools

In the section, we briefly describe some of the tools from the cryptography literature that we will be using in our protocols. We will base our description on the one given in [Pin03]. Good sources for further details are Ronald Cramer's lecture notes that provide an elementary introduction to the methods of secure computation [Cra00], and Oded Goldreich's manuscript detailing a rigorous introduction to secure multi-party computation [Gol98].

### 3.2.1 Oblivious Transfer

Oblivious transfer is a basic protocol, that is the main building block of secure computation. In fact, Kilian [Kil88] showed that oblivious transfer is sufficient for secure computation in the sense that given an implementation of oblivious transfer, one can construct any secure computation protocol without using any other cryptographic primitive.

The notion of 1-out-of-2 oblivious transfer was suggested by Even, Goldreich and Lempel [EGL85] as a variant of a different but equivalent type of oblivious transfer that has been suggested by Rabin [Rab81]. The protocol involves two parties – the *sender* and the *receiver*. The sender's input is a pair $(x_0, x_1)$ and the receiver's input is a bit $\sigma \in 0, 1$. At the end of the protocol, the receiver learns $x_\sigma$ (and nothing else), and the sender learns nothing. In other words, if we use the notation $(input_A, input_B) \rightarrow (output_A, output_B)$ to define the outcome of the function, then oblivious transfer is the function $((x_0, x_1), \sigma) \rightarrow (\lambda, x_\sigma)$, where $\lambda$ is the empty output.

It is known how to design oblivious transfer protocols based on virtually all known constructions of trapdoor functions, i.e. public key cryptosystems. For the case of semi-honest adversaries, there exist simple and efficient protocols for oblivious transfer [EGL85,

Gol98]. One straightforward approach is for the receiver to generate two random public keys – a key $P_\sigma$ whose decryption key he knows, and a key $P_{1-\sigma}$ whose decryption key he does not know. The receiver then sends these two keys to the sender, who encrypts $x_0$ with the key $P_0$ and encrypts $x_1$ with the key $P_1$, and sends the two encrypted values to the receiver. The receiver can then decrypt $x_\sigma$ but not $x_{1-\sigma}$. It is easy to show that the sender does not learn anything about $\sigma$, since the only message that she receives contains two random public keys, and there is no way for her to deduce the one for which the receiver has a private key. As for the sender's privacy, if the receiver follows the protocol, he knows the private key corresponding to only one of the public keys and can therefore decrypt only one of the inputs, and if the encryption scheme is secure he cannot gain information about the other input. To be secure against malicious adversaries, the oblivious transfer protocol must also ensure that the receiver chooses the public keys appropriately, i.e. the receiver knows the private key corresponding to only one of the public keys. This can be done by using zero-knowledge proofs by which the receiver proves that he has chosen the keys correctly. Fortunately, there are very efficient zero-knowledge proofs for this case, see e.g. [NP01]. Oblivious transfer is often the most computationally intensive operation of secure protocols, and is repeated many times. Each invocation of oblivious transfer typically requires a constant number of invocations of trapdoor permutations (i.e. public-key operations, or exponentiations). It is possible to reduce the amortized overhead of $n$ oblivious transfers to one exponentiation per $\log n$ oblivious transfers, even for the case of malicious adversaries [NP01].

### 3.2.2 Yao's Protocol for Two-party Secure Computation

In [Yao86], Yao presented a constant-round protocol for privately computing any probabilistic polynomial-time function. Denote the parties as Alice (A) and Bob (B), and denote their respective inputs by $x$ and $y$. Let $f$ be the function that they wish to compute (for simplicity, we assume that we want only Bob to learn the value of $f(x, y)$ at the end of the protocol). The protocol is based on expressing $f$ as a combinatorial circuit with gates defined over some fixed base $G$. For example, $G$ can include all the functions $g : \{0, 1\} \times \{0, 1\} \to \{0, 1\}$. The bits of the input are entered into input wires and are

propagated through the gates. Note that it is known that any polynomial-time function can be expressed as a combinatorial circuit of polynomial size (see, e.g. [Sav72]).

**Encoding the Circuit.** Informally, Yao's protocol works by having one of the parties (say Alice) first generate an *encrypted* or *garbled* circuit for computing $f$ and send its representation to Bob. The encrypted circuit is generated in the following way: first, Alice *hardwires* her input into the circuit, generating a circuit computing $f(x, \cdot)$. She then assigns to each wire $i$ of the circuit two random "garbled" values $(W_0^i; W_1^i)$ corresponding to values $0$ and $1$ of the wire (the random values should be long enough to be used as keys to a pseudo-random function, e.g. 80-128 bits long). Consider a gate $g$ which computes the value of the wire $k$ as a function of wires $i$ and $j$. Alice prepares a table $T_g$ that encrypts the garbled value of the output wire using the output of a pseudo-random function $F$ keyed by the garbled values of the input wires $i$ and $j$. The table therefore has four entries, one entry for every combination of input values. (Note that pseudo-random functions are usually realized using private-key primitives such as block ciphers or hash functions, and are therefore very efficient.) The table enables computation of the garbled output of $g$, from the garbled inputs to $g$. Moreover, given the two garbled inputs to $g$, the table does not disclose information about the output of $g$ for any other inputs, nor does it reveal the values of the actual input or output bits. The representation of the circuit includes the wiring of the original circuit (namely, a mapping from inputs or gate outputs to gate inputs), the tables $T_g$, and tables that translate the garbled values of the output wires of the circuit to actual 0/1 values. In this form, the representation reveals nothing but the wiring of the circuit, and therefore Bob learns nothing from this stage. (We assume that the function $f$ is public and the wiring of the circuit is not secret).

**Encoding Bob's Input.** The tables described above enable the computation of the garbled output of every gate from its garbled inputs. Therefore, if Bob is given the garbled values of the input wires of the circuit in addition to these tables, he can compute the garbled values of its output wires and then translate them to actual values. In order for Bob to obtain the garbled values of the input wires, Alice and Bob engage, for each input wire, in a 1-out-of-2 oblivious transfer. In this protocol, Alice is the sender, and her input pair

consists of the two garbled values of this wire, and Bob is the receiver, and his input is his input bit. As a result of the oblivious transfer protocol Bob learns the garbled value of his input bit and nothing about the garbled value of the other bit, and Alice learns nothing.

**Computing the Circuit.** At the end of the oblivious transfer stage, Bob has sufficient information to compute the output of the circuit on his own. After computing $f(x, y)$, he can send this value to Alice if she requires it.

**Security of the Protocol.** To show that the protocol is secure, it must be shown that none of the two parties learns anything that it cannot compute from the input and output only. To see that the protocol is secure against Alice, we note that Alice receives messages only during the oblivious transfer protocol, and being the sender, she does not learn anything from that protocol. As for security against Bob, we observe that every masking value (e.g. the output of the pseudo-random function $F$) is used only once, and that the pseudo-randomness of $F$ ensures that without knowledge of the correct keys, i.e. garbled values of the input wires of a gate, the output values of the gate look random. Therefore, knowledge of one garbled value of each of the input wires of a gate discloses only a single garbled output value of the gate and Bob cannot distinguish the other garbled value from random. Now, the oblivious transfer protocol ensures that Bob learns only a single garbled value for each input wire. Therefore, inductively, Bob can compute only a single garbled value of each gate, and in particular of the output of the circuit. Moreover, the method by which the tables were constructed hides the values of intermediate results (i.e. of gate outputs inside the circuit).

**Overhead.** The protocol involves: (1) Alice and Bob engaging in an oblivious transfer protocol for every input wire of the circuit, (2) Alice sending to Bob tables of size linear in the size of the circuit, and (3) Bob computing a pseudo-random function a constant number of times for every gate (this is the cost incurred in evaluating the gates). The number of rounds of the protocol is constant (namely, two rounds using the oblivious transfer of [EGL85, Gol98, NP01]). The computation overhead is dominated by the oblivious transfer stage, since the evaluation of the gates uses pseudo-random functions which are very

efficient compared to the oblivious transfer protocol. It is roughly linear in the size of Bob's input. The communication overhead, on the other hand, is linear in the size of the circuit. Thus, Yao's protocol is efficient if and only if the circuit representation of $f$ and the input of one of the parties are not very large.

### 3.2.3 Secure Comparison

The main primitive that we use in our two-party protocol is a protocol for secure comparison. The protocol involves two parties – party A having an input $x$ and party B having an input $y$. The output is 0 if $x \geq y$ and 1 otherwise. We can implement the protocol by encoding the comparison function as a binary circuit and then using Yao's protocol for secure computation. The overhead is $|x|$ oblivious transfers, and $O(|x| + |y|)$ applications of a pseudo-random function, as well as $O(|x| + |y|)$ communication. More efficient, non-interactive comparison protocols also exist (see e.g. [Fis01]). Among other specialized protocols for this problem is a protocol suggested by Cachin that ensures fairness given a semi-trusted third party [ACCK01].

### 3.2.4 Reactive Computation

A reactive computation consists of a series of steps in which parties provide inputs and receive outputs. Each step generates a *state* which is used by the following step. The input that a party provides at step $i$ can depend on the outputs that it received in previous steps. (We limit ourselves to synchronous communication, and to an environment in which there are secure channels between the parties.) The protocols that we design for the malicious case implement reactive computation. We refer the reader to [Can01, CLOS02] for a discussion of the security definitions and constructions for reactive computation.

### 3.2.5 A Composition Theorem

Our protocols implement the computation of the $k^{th}$-ranked element by running many invocations of secure computation of simpler functionalities. Such constructions are covered by theorems of secure composition [Can00, Can01]. Loosely speaking, consider a *hybrid*

*model* where the protocol uses a trusted party that computes the functionalities $f_1, \ldots, f_\ell$. The secure composition theorem states that if we consider security in terms of comparing the real computation to the ideal model, then if a protocol is secure in the hybrid model, and we replace the calls to the trusted party by calls to secure protocols computing $f_1, \ldots, f_\ell$, then the resulting protocol is secure. A secure composition theorem applies to reactive computation, too [Can01, CLOS02].

## 3.3 Two-party Computation of the $k^{th}$-ranked Element

This section describes protocols for secure two-party computation of the $k^{th}$-ranked element of the union of two databases. The protocols are based on the observation that a natural algorithm for computing the $k^{th}$-ranked element discloses very little information that cannot be computed from the value of the $k^{th}$-ranked element itself. Some modification to that protocol can limit the information that is leaked by the execution to information that can be computed from the output alone.

To simplify the description of the basic insecure protocol, we describe it for the case of two parties, A and B, each having an input of size $n/2$, that wish to compute the value of the median, i.e. $(n/2)^{th}$-ranked element, of the union of their two inputs sorted in increasing order of their values. This protocol is a modification of the algorithm given in [Rod82, KN97]. [3] Assume for simplicity that all input values are different. The protocol operates in rounds. In each round, each party computes the median value of his or her remaining input, and then the two parties compare their two median values. If A's median value is smaller than B's then A adjusts her input by removing the values which are less than or equal to her median, and B removes from his input items which are greater than his median. Otherwise, A removes from her input items which are greater than her median and B removes from his input items which are less than or equal to his median. The protocol continues until the remaining input sets are of length 1 (thus the number of rounds is logarithmic in the number of input items). The protocol is correct since when A's median

[3] Another variant of the algorithm that is presented there, and is due to Karchmer, reduces the communication overhead to $O(\log n)$ bits (instead of $O(\log^2 n)$). Our protocols do not use this improvement. In any case, the communication associated with the secure computation overshadows the communication overhead of the basic protocol.

is smaller than B's median, each of the items that A removes is smaller than A's median, which is smaller than at least $n/4$ inputs of A and $n/4+1$ inputs of B. Therefore, the items removed by A are all smaller than the median. Similarly, all the items removed by B are larger than the median. Moreover, the protocol removes $n/4$ items smaller than the median and $n/4$ items greater than it; therefore, the median of the new data is the same as that of the original input. The other case follows similarly.

Suppose now that the comparison is done privately, i.e. the parties only learn which party's median value is greater, and do not learn any other information about each other's median value. We show below that, in this case, the protocol is secure. Intuitively, this is true because each party can deduce the result of the comparison from the value of the overall median and its own input. For example, if party A knows the median value of her input and the median of the union of the two inputs, and observes that her median is smaller than the median of the union, then she can deduce that her median value is smaller than that of B. This means that given the final output of the protocol, both parties can simulate the results of the comparisons. Consequently, we have a reduction from the problem of securely computing the median of the union to the problem of securely computing comparisons.

### 3.3.1 Protocol for Semi-Honest and Malicious Parties

Following is a description of a protocol that finds the $k^{th}$-ranked element in the union of two databases and is secure against semi-honest parties. The computation of the median is a specific case where $k$ is set to be the sum of the sizes of the two inputs divided by two. The protocol reduces the general problem of computing the $k^{th}$-ranked element of arbitrary size inputs, to the problem of computing the median of two inputs of equal size, which is also a power of 2. For now, we will assume that all the inputs are *distinct*. This issue is further discussed later.

**Security Against a Malicious Adversary.** The protocol for the semi-honest case can be amended to be secure against malicious adversaries. The main change is that the protocol must now verify that the parties provide consistent inputs to the different invocations of the secure computation of the comparisons. For example, if party A gave an input of value $100$

to a secure comparison computation, and the result was that A must delete all its input items which are smaller than $100$, then $A$ cannot provide an input which is smaller than $100$ to any subsequent comparison. We provide a proof that given this enforcement, the protocol is secure against malicious behavior. For this protocol, we do not force the input elements to be integers. However, if such an enforcement is required (e.g. if the input consists of rounded salary data), then the protocol for the malicious case also verifies that there is room for sufficiently many distinct integers between the reported values of different elements of the input. This is made more precise later.

In protocol FIND-RANKED-ELEMENT that we describe here, we also specify the *additional functionality* that is required in order to ensure security against malicious parties. Then in Section 3.3.3 we describe how to implement this functionality, and show that given this functionality, the protocol is secure against malicious adversaries. Of course, to obtain a protocol which is secure only against semi-honest adversaries, one should ignore the additional highlighted steps that provide security in the malicious case.

**Protocol** FIND-RANKED-ELEMENT
**Input:** $D_A$ known to A, and $D_B$ known to B. Public parameter $k$ (for now, we assume that the numerical value of the rank of the element is known). All items in $D_A \cup D_B$ are distinct.
**Output:** The $k^{th}$-ranked element in $D_A \cup D_B$.

1. Party A (resp., B) initializes $S_A$ (resp., $S_B$) to be the sorted sequence of its $k$ smallest elements in $D_A$ (resp., $D_B$).
2. If $|S_A| < k$ then Party A pads $(k - |S_A|)$ values of "+$\infty$" to its sequence $S_A$. Party B does the same: if $|S_B| < k$ then it pads $(k - |S_B|)$ values of "+$\infty$" to its sequence $S_B$.
3. Let $2^j$ be the smallest power of 2 greater than or equal to $k$. Party A pre-pads $S_A$ with $(2^j - k)$ values of "-$\infty$" and Party B pads $S_B$ with $(2^j - k)$ values of "+$\infty$". (The result is two input sets of size $2^j$ each, whose median is the $k^{th}$-ranked element in $D_A \cup D_B$ .)

   **In the malicious case:** The protocol sets bounds $l_A = l_B = -\infty$ and $u_A = u_B = \infty$.

4. For $i = (j-1), \ldots, 0$ : [Round $i$]

   A. A computes the $(2^i)^{th}$ element of $S_A$, denoted $m_A$, and B computes the $(2^i)^{th}$ element of $S_B$, $m_B$. (I.e., they compute the respective medians of their sets.)

   B. A and B engage in a *secure computation* which outputs 0 if $m_A \geq m_B$, and 1 if $m_A < m_B$.
   **In the malicious case:** The secure computation first checks that $l_A < m_A < u_B$ and $l_B < m_B < u_B$. If we want to force the input to be integral, then we also check that $l_A + 2^i \leq m_A \leq u_A - 2^i$ and $l_B + 2^i \leq m_B \leq u_B - 2^i$. If these conditions are not satisfied, then the protocol is aborted. Otherwise, if $m_A \geq m_B$, the protocol sets $u_A = m_A$ and $l_B = m_B$. Otherwise it updates $l_A$ to $m_A$ and $u_B$ to $m_B$. Note that the lower and upper bounds are not revealed to either party.

   C. If $m_A < m_B$, then A removes all elements ranked $2^i$ or less from $S_A$, while B removes all elements ranked greater than $2^i$ from $S_B$. On the other hand, if $m_A \geq m_B$, then A removes all elements ranked higher than $2^i$ from $S_A$, while B removes all elements ranked $2^i$ or lower from $S_B$.

5. (By now, both $S_A$ and $S_B$ are of size 1.) Party $A$ and $B$ output the result of a *secure computation* which computes the smaller of the two remaining elements.

   **In the malicious case:** The secure computation checks that the inputs given in this step are consistent with the inputs given earlier. Specifically, for any item other than item $2^j$ of the original set of A (respectively B), this means that the value must be equal to $u_A$ (respectively $u_B$). For the item ranked $2^j$ in the original set of party A (respectively B), it is verified that its value is greater than $l_A$ (respectively $l_B$).

**Overhead:** Since the value $j$ is at most $\log 2k$ and the number of rounds of communication is $(j+1)$, the total number of rounds of communication is at most $\log 2k$. In each round, the protocol performs at most one secure computation, which requires a comparison of $(\log M)$-bit integers. Thus the total communication cost is $O(\log M \cdot \log k)$ times the security parameter.

**Proof of Correctness**

Regardless of security issues, we first have to show that the protocol indeed computes the $k^{th}$-ranked item. We need to show that (a) The preprocessing performed in Steps 1-3 does not eliminate the $k^{th}$-ranked element and (b) The $(2^{i+1})^{st}$ smallest value of $S_A^i \cup S_B^i$ is the $k^{th}$-ranked element in $D_A \cup D_B$ for each $i = j-1, \ldots, 0$ (where $S_A^i, S_B^i$ are the sorted sequences maintained by parties $A$, $B$, respectively, during round $i$). These two properties are shown in Lemma 3.3.1.

**Lemma 3.3.1** *In Protocol* FIND-RANKED-ELEMENT, *the $(2^{i+1})^{st}$-ranked element (i.e., the median) of $S_A \cup S_B$ in round $i$ of Step 4 is equal to the $k^{th}$-ranked element in $D_A \cup D_B$, for $i = (j-1), \ldots, 0$.*

**Proof:** Note that in the preprocessing (Step 1) we do not eliminate the $k^{th}$-ranked element since the $k^{th}$-ranked element cannot appear in position $(k+1)$ or higher in the sorted version of $D_A$ or $D_B$. Step 2 ensures that both sequences have size exactly $k$ without affecting the $k^{th}$-ranked element (since padding is performed at the end of the sequences). And, Step 3 not only ensures that the length of both sequences is a power of 2, but also pads $S_A$ and $S_B$ so that the $(2^j)^{th}$ element of the union of the two sequences is the $k^{th}$-ranked element of $D_A \cup D_B$. This establishes the lemma for the case where $i = (j-1)$.

The remaining cases of $i$ follow by induction. By induction hypothesis, at the beginning of round $i$, the original problem is equivalent to the problem of computing the median between two sets of size $2^{i+1}$. We claim that, in round $i$, neither party removes the median of $S_A \cup S_B$: if $m_A < m_B$ then there are $2 \cdot 2^i + 1$ points in $S_A$ and $S_B$ that are larger than $m_A$ and $2 \cdot 2^i - 1$ points in $S_A$ and $S_B$ that are smaller than $m_B$; thus all points in $S_A$ that are less than or equal to $m_A$ are smaller than the median, and all points in $S_B$ that are greater than $m_B$ are greater than the median. A similar argument follows in the case

that $m_A > m_B$. Furthermore, the modifications made to $S_A$ and $S_B$ maintain the median of $S_A \cup S_B$ since in each round, an equal number of elements smaller than and greater than the median are removed. The lemma follows. □

### 3.3.2 Security for the Semi-honest Case

In the semi-honest case, the security definition in the ideal model is identical to the definition which is based on simulation. Thus, it is sufficient to show that, assuming that the number of elements held by each party is public information, party A (and similarly party B), given its own input and the value of the $k^{th}$-ranked element, can simulate the execution of the protocol in the hybrid model, where the comparisons are done by a trusted party; then the security of the protocol follows from the composition theorem. We describe the proof details for the case of party A simulating the execution in the hybrid model. Let $x$ be the $k^{th}$-ranked element which the protocol is supposed to find. Then, party A simulates the protocol as follows:

**Algorithm** SIMULATE-FIND-RANK
**Input:** $D_A$ and $x$ known to A. Public parameter $k$. All items in $D_A \cup D_B$ are distinct.
**Output:** Simulation of running the protocol for finding the $k^{th}$-ranked element in $D_A \cup D_B$.

1. Party A initializes $S_A$ to be the sorted sequence of its $k$ smallest elements in $D_A$.
2. If $|S_A| < k$ then Party A pads $(k - |S_A|)$ values of "+$\infty$" to its sequence $S_A$.
3. Let $2^j$ be the smallest power of 2 larger than $k$. Party A pre-pads $S_A$ with $(2^j - k)$ values of "-$\infty$".
4. For $i = (j-1), \ldots, 0:$ [Round $i$]
   A. A computes $m_A$, the $(2^i)^{th}$ element of $S_A$.
   B. If $m_A < x$, then the secure comparison is made to output 1, i.e., $m_A < m_B$, else it outputs 0.

C. If $m_A < x$, then A removes all elements ranked $2^i$ or less from $S_A$. On the other hand, if $x \leq m_A$, then A removes all elements ranked higher than $2^i$ from $S_A$.

5. The final secure computation outputs $x$.

**Lemma 3.3.2** *The transcript generated by Algorithm* SIMULATE-FIND-RANK *is the same as the transcript generated by Protocol* FIND-RANKED-ELEMENT. *In addition, the state information that Party A has after each round, namely ($S_A$, $k$), correctly reflects the state of Protocol* FIND-RANKED-ELEMENT *after the same round.*

**Proof:** We prove the lemma by induction on the number of rounds. Assume that the lemma is true at the beginning of a round, i.e. Algorithm SIMULATE-FIND-RANK has been correctly simulating Protocol FIND-RANKED-ELEMENT and its state correctly reflects the state of Protocol FIND-RANKED-ELEMENT at the beginning of the round. We show that $m_A < x$ if and only if $m_A < m_B$. If $m_A < x$, then the number of points in $S_A^i$ smaller than $x$ is at least $2^i$. If, by way of contradiction, $m_B \leq m_A$, then $m_B < x$, implying that the number of points in $S_B^i$ smaller than $x$ is at least $2^i$. Thus the total number of points in $S_A^i \cup S_B^i$ smaller than $x$ would be at least $2^{i+1}$, contradicting that $x$ is the median. So, $m_A < m_B$. On the other hand, if $m_A < m_B$, and by way of contradiction, $m_A \geq x$, then $x \leq m_A < m_B$, implying that the number of points in $S_B^i$ greater than $x$ is strictly more than $2^i$. Also, at least $2^i$ points in $S_A^i$ are greater than $x$. Thus, the number of points in $S_A^i \cup S_B^i$ greater than $x$ is strictly more than $2^{i+1}$, again contradicting that $x$ is the median. So, $m_A < x$. Therefore, the secure comparisons in Step 4 of Algorithm SIMULATE-FIND-RANK return the same outputs as in Protocol FIND-RANKED-ELEMENT, thereby ensuring that the set $S_A$ is also updated correctly. □

**Duplicate Items.** Protocol FIND-RANKED-ELEMENT preserves privacy as long as no two input elements are identical (this restriction must be met for each party's input, and also for the union of the two inputs). The reason for this restriction is that the execution of the protocol reveals to each party the exact number of elements in the other party's input which are smaller than the $k^{th}$-ranked item of the union of the two inputs. If all elements are distinct then, given the value of the $k^{th}$-ranked element, each party can compute the

number of elements in its own input that are smaller than it, and therefore each party can also compute the number of such elements in the other party's input. This information is sufficient for simulating the execution of the protocol. However, if the input contains identical elements then, given the value of the $k^{th}$-ranked element, it is impossible to compute the exact number of elements in the other party's input which are smaller than it, thus preventing one from simulating the protocol. For example, if several items in A's input are equal to the $k^{th}$-ranked element then the protocol could have ended with a comparison involving any one of them. Therefore A does not know which of the possible executions took place.

**Handling Duplicate Items.** Protocol FIND-RANKED-ELEMENT-MULTIPARTY in Section 3.4 can securely compute the $k^{th}$-ranked item even if the inputs contain duplicate elements, and can be applied to the two-party case (the downside is that it involves $\log M$ rounds, instead of $\log k$). Also, protocol FIND-RANKED-ELEMENT can be applied to inputs that might contain identical elements, if they are transformed into inputs containing distinct elements. This can be done, for example, in the following way: Let the total number of elements in each party's input be $n$. Add $\lceil \log n \rceil + 1$ bits to every input element, in the least significant positions. For every element in A's input, let these bits be a "0" followed by the rank of the element in a sorted list of A's input values. Apply the same procedure to B's inputs using a "1" instead of a "0". Now run the original protocol using the new inputs, but ensure that the output does not include the new least significant bits of the $k^{th}$-ranked item. The protocol is privacy-preserving with regard to the new inputs (which are all distinct). Also, this protocol does not reveal to party A more information than running the original protocol with the original inputs and providing A with the additional information of the number of items in B's input which are smaller than the $k^{th}$-ranked element (the same holds of course w.r.t. B as well). This property can be verified by observing that if A is given the $k^{th}$-ranked element of the union of the two inputs, as well as the number of elements in $B$'s input which are smaller than this value, it can simulate the operation of the new protocol with the transformed input elements.

**Hiding the Size of the Inputs.** We now consider the case where the parties wish to hide the size of their inputs from each other. Note that if $k$ is public then the protocol that we described already hides the sizes of the inputs, since each party transforms its input to one of size $k$. This solution in insufficient, though, if $k$ discloses information about the input sizes. For example, if the protocol computes the median, then $k$ is equal to half the sum of the sizes of the two inputs. We next show how to hide the size of the inputs when the two parties wish to compute the value of the element with rank $\lfloor \phi n \rfloor$.

Let $\phi = \phi_n/\phi_d$, where both $\phi_n$ and $\phi_d$ are integers. We assume $U$, a multiple of $\phi_d$, is known to be an upper bound on the number of elements held by each party. If both $S_A$ and $S_B$ divide $\phi_d$, then party A pads its input with $\phi(U - |S_A|)$ elements with value $-\infty$, and $(1-\phi)(U - |S_A|)$ elements with value $+\infty$; similarly, party B pads its input with $\phi(U - |S_B|)$ elements with value $-\infty$, and $(1-\phi)(U - |S_B|)$ elements with value $+\infty$. Otherwise, party $X$, for $X \in \{A, B\}$, needs to reveal the value of $|S_X| \bmod \phi_d$ to the other party. We note that for small values of $\phi_d$, such a revelation is usually acceptable even in cases where the two parties want to hide the size of their respective data sets from each other. Let $r_A = \phi_d - (|S_A| \bmod \phi_d)$ and let $r_B = \phi_d - (|S_B| \bmod \phi_d)$. First, party A adds $\phi(U - (|S_A| + r_A))$ elements with value $-\infty$ and $(1-\phi)(U - (|S_A| + r_A))$ elements with value $+\infty$; similarly party B adds $\phi(U - (|S_B| + r_B))$ elements with value $-\infty$ and $(1-\phi)(U - (|S_B| + r_B))$ elements with value $+\infty$. Next, assume without loss of generality that $r_A \geq r_B$ (otherwise, interchange the role of A and B in the following). If $\phi \leq 0.5$ then party A adds $\lceil \phi(r_A + r_B) \rceil$ more elements with value $-\infty$ and adds $r_A - \lceil \phi(r_A + r_B) \rceil$ more elements with value $+\infty$, while party B adds $r_B$ more elements with value $+\infty$. If $\phi > 0.5$, then party A adds $\lfloor (1-\phi)(r_A + r_B) \rfloor$ more elements with value $\infty$ and adds $r_A - \lfloor (1-\phi)(r_A + r_B) \rfloor$ more elements with value $-\infty$, while party B adds $r_B$ more elements with value $-\infty$. Then the parties engage in a secure computation of the $(2\phi U)^{th}$-ranked element of the new inputs, using the protocol we described above.

### 3.3.3 Security for the Malicious Case

We assume that the comparison protocol is secure against malicious parties. We then show that although the malicious party can choose its input values adaptively during the execution of the protocol, it could as well have constructed an input a priori and given it to a trusted third party to get the same output. In other words, although the adversary can define the values of its input points depending on whether that input point needs to be compared or not in our protocol, this does not give it any more power. The proof is composed of two parts. First, we show that the functionality provided by the protocol definition provides the required security. Then, we show how to implement this functionality.

**Lemma 3.3.3** *For every adversary $A'$ in the real model there is an adversary $A''$ in the ideal model, such that the outputs generated by $A'$ and $A''$ are computationally indistinguishable.*

**Proof:** Based on the composition theorem, we can consider a protocol in the hybrid model where we assume that the comparisons are done securely by a trusted party. (We actually need a composition theorem for a reactive scenario here. We refer the reader to [Can01, CLOS02] for a treatment of this issue.)

Visualize the operation of $A'$ as a binary tree. The root is its input to the first comparison performed in the protocol. The left child of the root is its input to the second comparison if the answer to the first comparison is 0, and the right child is its input to the second comparison if the first answer is 1. The tree is constructed recursively following this structure, where every node corresponds to the input provided by $A'$ to a comparison done at Step 4(B). We add leaves corresponding to the input provided by $A'$ to the secure computation in Step 5 of the protocol; note that for each possible outcome of the sequence of comparisons in Step 4(B), there is a unique leaf corresponding to it.

Fix the random input used by adversary $A'$. We also limit ourselves to adversaries that provide inputs within the bounds maintained by the protocol (otherwise the protocol aborts, and since early termination is legitimate in the ideal model, we are done). We must generate an input that can be given to the trusted party in the ideal model in order to generate a computationally-indistinguishable transcript. For this, we run $A'$ by providing it with the output of the comparisons. We go over all execution paths (i.e. paths in the tree) by stopping

and rewinding the operation. (This is possible since the tree is of logarithmic depth.) Note that each of the *internal* nodes corresponds to a comparison involving a *different* location in the sorted list that $A'$ is supposed to have as its input. Associate with each node the value that $A'$ provides to the corresponding comparison. Observe the following facts:

- For any three internal nodes $L, A, R$ where $L$ and $R$ are the left and right children of $A$, the bounds checked by the protocol enforce that the value of $L$ is smaller than that of $A$, which is smaller than that of $R$. Furthermore, an inorder traversal of the internal nodes of the tree results in a list of distinct values appearing in ascending order.

- When the computation reaches a leaf (Step 5), $A'$ provides a single value to a comparison. For the rightmost leaf, the value is larger than any value seen till now, while for each of the remaining leaves, the value is the same as the value on the rightmost internal node on the path from the root to the leaf (this is enforced by checking that the value is the same as $u_A$ or $u_B$ respectively).

- Each item in the input of $A'$ is used in at most a single internal node and exactly one leaf of the tree.

Consequently, the values associated with the leaves are sorted, and agree with all the values that $A'$ provides to comparisons in the protocol. We therefore use these values as the input to the trusted third party in the ideal model. When we receive the output from the trusted party we simulate the route that the execution takes in the tree, provide outputs to $A'$ and $B$, and perform any additional operation that $A'$ might apply to its view in the protocol.[4] □

**Implementing the Functionality of the Malicious-Case Protocol**

The functionality that is required for the malicious case consists of using the results of the first $i$ comparisons to impose bounds on the possible inputs to the following comparison. This is a *reactive secure computation*, which consists of several steps, where each step

[4]Note that we are assuming that the inputs can be arbitrary real numbers. If, on the other hand, there is some restriction on the form of the inputs, the protocol must verify that $A'$ provides values which are consistent with this restriction. For example, if the inputs are integers then the protocol must verify that the distance between the reported median and the bounds is at least half the number of items in the party's input (otherwise the input items cannot be distinct).

operates based on inputs from the parties and state information that is delivered from the previous step. This scenario, as well as appropriate security definitions, was described in [Can01, CLOS02].

Reactive computation can be implemented using the generic constructions provided in [Can01, CLOS02], which are secure against malicious parties. The main idea there is to construct a separate circuit for evaluating each step, where part of the output of every circuit is the state information which is input to the next circuit. This state information is output in an encrypted and authenticated form, to prevent the party evaluating the circuit from learning or changing it. It must be verified and decrypted by the secure computation of the following step. We can use efficient encryption and authentication schemes, like one-time pads and universal hashing, for this purpose.

## 3.4 Multi-party Computation of the $k^{th}$-ranked Element

We now describe a protocol that outputs the value of the $k^{th}$-ranked element of the union of multiple databases. For this protocol we assume that the elements of the sets are integer-valued, but they need not be distinct. Let $[\alpha, \beta]$ be the (publicly-known) range of input values, and let $M = \beta - \alpha + 1$. The protocol runs a series of rounds in which it (1) suggests a value for the $k^{th}$-ranked element, (2) performs a secure computation to which each party reports the number of its inputs which are smaller than this suggested value, adds these numbers and compares the result to $k$, and (3) updates the guess. The number of rounds of the protocol is logarithmic in $M$.

**Malicious Adversaries.** We describe a protocol which is secure against semi-honest adversaries. The protocol can be amended to be secure against malicious adversaries by verifying that the parties are providing it with consistent inputs. We specify in the protocol the additional functionality that needs to be implemented in order to provide security against malicious adversaries.

**Protocol** FIND-RANKED-ELEMENT-MULTIPARTY

**Input:** Party $P_i$, $1 \leq i \leq s$, has database $D_i$. The sizes of the databases are public, as is

the value $k$. The range $[\alpha, \beta]$ is also public.
**Output:** The $k^{th}$-ranked element in $D_1 \cup \cdots \cup D_s$.

1. Each party ranks its elements in ascending order. Initialize the current range $[a, b]$ to $[\alpha, \beta]$ and set $n = \sum |D_i|$.
   **In the malicious case:** Set for each party $i$ bounds $(l)^i = 0,\ (g)^i = 0$. These values are used to bound the inputs that party $i$ reports in the protocol. $(l)^i$ reflects the number of inputs of party $i$ strictly smaller than the current range, while $(g)^i$ reflects the number of inputs of party $i$ strictly greater than the current range.
2. Repeat until "done"
   (a) Set $m = \lceil (a+b)/2 \rceil$ and announce it.
   (b) Each party computes the number of elements in its database which are strictly smaller than $m$, and the number of elements strictly greater than $m$. Let $l_i$ and $g_i$ be these values for party $i$.
   (c) The parties engage in the following secure computation:
   **In the malicious case:** Verify for every party $i$ that $l_i + g_i \leq |D_i|$, $l_i \geq (l)^i$, and $g_i \geq (g)^i$. In addition, if $m = a$, then we check that $l_i = (l)^i$; or if $m = b$, we verify that $g_i = (g)^i$.
   - Output "done" if $\sum l_i \leq k-1$ and $\sum g_i \leq n-k$. (This means that $m$ is the $k^{th}$-ranked item.)
   - Output "0" if $\sum l_i \geq k$. In this case, set $b = m-1$. (This means that the $k^{th}$-ranked element is smaller than $m$.)
     **In the malicious case:** Set $(g)^i = |D_i| - l_i$. (Note that as the right end-point of the range decreases, $(g)^i$ is non-decreasing. This can be seen by noting that $|D_i| - l_i \geq g_i$, which is enforced to be at least as much as the previous value of $(g)^i$. (Since the left end-point of the range remains the same, $(l)^i$ remains unchanged.)
   - Output "1" if $\sum g_i \geq n-k+1$. In this case set $a = m+1$. (This means that the $k^{th}$-ranked element is larger than $m$.)
     **In the malicious case:** Set $(l)^i = |D_i| - g_i$.

**Correctness:** The correctness of this algorithm follows from observing that if $m$ is the $k^{th}$-ranked element then the first condition will be met and the algorithm will output it. In the other two cases, the $k^{th}$-ranked element is in the reduced range that the algorithm retains.

**Overhead:** The number of rounds is $\log M$. Each round requires a secure multi-party computation that computes two summations and performs two comparisons. The size of the circuit implementing this computation is $O(s \log M)$, which is also the number of input bits. The secure evaluation can be implemented using the protocols of [GMW87, BMR90, FY92].

### 3.4.1 Security for the Semi-honest Case

We provide a sketch of the proof of the security of the protocol. Assume that the multi-party computation in step 2(c) is done by a trusted party. Denote this scenario as the hybrid model. We show that in this case the protocol is secure against an adversary that controls up to $s-1$ of the parties. Now, if we implement the multi-party computation by a protocol which is secure against an adversary that controls up to $t$ parties, e.g. using [GMW87, BMR90, FY92], it follows from the composition theorem that the resulting protocol is secure against this adversary. Of course, $t < s-1$ in the actual implementation, since the protocols computing the "simple" functionalities used in the hybrid model are not secure against $s-1$ parties, but rather against, say, any coalition of less than $s/3$ corrupt parties.

In the hybrid model, the adversary can simulate its view of the execution of the protocol, given the output of the protocol (and without even using its input). Indeed, knowing the range $[a, b]$ that is used at the beginning of a round, the adversary can compute the target value $m$ used in that round. If $m$ is the same as the output, it concludes that the protocol must have ended in this round with $m$ as the output (if the real execution did not output $m$ at this stage, $m$ would have been removed from the range and could not have been output). Otherwise, it simply updates the range to that side of $m$ which contains the output (if the real execution had not done the same, the output would have gone out of the active range and could not have been the output). Along with the knowledge of the initial range, this

shows that the adversary can simulate the execution of the protocol.

### 3.4.2 Security for the Malicious Case

We show that the protocol is secure given a secure implementation of the functionality that is described in Step 3 of algorithm FIND-RANKED-ELEMENT-MULTIPARTY. Since this is a multi-party reactive system we refer the reader to [Can01, CLOS02] for a description of such a secure implementation. The idea is that the parties run a secure computation of each step using, e.g., the protocol of [BMR90]. The output contains encrypted and authenticated *shares* of the current state, which are then input to the computation of the following step, and checked by it.

For every adversary that corrupts up to $s-1$ parties in the computation in the hybrid model, there is an adversary with the same power in the ideal model. We limit the analysis to adversaries that provide inputs that agree with all the boundary checks in the algorithm (otherwise the protocol aborts, and this is a legitimate outcome in the ideal model).

Imagine a tree of size $M$ with each node in the tree corresponding to a guess $m$ (in the protocol) for the value of the median. The root corresponds to the initial guess $m = m_0 = \lceil(\beta-\alpha)/2\rceil$ with the initial range being $[\alpha, \beta]$. Its left child corresponds to the next guess for $m$ if the first guess is incorrect and the median is discovered to be smaller than $m_0$ and is associated with the range $[a, b]$, with $a = \alpha$ and $b = m_0 - 1$. Similarly, the right child corresponds to the next guess for $m$ if the median is discovered to be larger than $m_0$, and is associated with the range $[m_0 + 1, \beta]$. The whole tree is constructed recursively in this manner. The leaves are associated with ranges containing a single integer. Note that each integer in the interval $[\alpha, \beta]$ is associated with the single node $u$ in the tree at which the guess $m$ is set to the value of this integer.

Fix the random values (coin flips) used by the adversary in its operation. Run the adversary, with rewinding, checking the values that are given by each of the parties it controls to each of the comparisons. Let the guess $u$ be associated with the node $\overline{u}$. Then, the two values $l_u$ and $g_u$ that party $p$ provides to the comparison of node $\overline{u}$ are supposed to be the number of items in the input of party $p$ which are smaller than and larger than $u$ respectively. Also, $e_u = |D_i| - l_u - g_u$ denotes the number of items that are specified

by the adversary to be equal to $u$. Assume that we first examine the adversary's behavior for the root node, then for the two children of the root, and continue layer by layer in the tree. The boundary checks ensure that if node $\overline{i+1}$ is a descendant of node $\bar{i}$, then $l_{i+1} = D_i - g_i = l_i + e_i$, and if node $\bar{i}$ is a descendant of node $\overline{i+1}$, then $g_i = D_{i+1} - l_{i+1}$, or equivalently that $l_i + e_i = l_{i+1}$. We next observe that by construction, for any consecutive nodes $\bar{i}$ and $\overline{i+1}$, either $\bar{i}$ is a descendant of $\overline{i+1}$ or $\overline{i+1}$ is a descendant of $\bar{i}$. This is so because only a node with value between $u$ and $v$ can put two nodes $\overline{u}$ and $\overline{v}$ into different subtrees. Thus, for $i = \alpha, \ldots, \beta - 1$, we have $l_i + e_i = l_{i+1}$. Moreover, our checks make sure than $l_\alpha = 0$ and $g_\beta = 0$ implying $l_\beta + e_\beta = D_i$. Summing over all these equalities, we get $\sum_{i=\alpha}^{\beta} e_i = D_i$.

Thus, we can use the result of this examination to define the input that a corrupt party $p$ provides to the trusted party in the ideal model. Specifically, we set the input to contain $e_u$ items of value $u$, for every $u \in [\alpha, \beta]$. The trusted party computes the $k^{th}$-ranked element, say using the same algorithms as in the protocol. Since in the protocol itself, the values provided by each party depend only on the results of previous comparisons (i.e. path in the tree), the output of the trusted party is the same as in the protocol.

# Part II

# Auctions for Web Advertisements

# Chapter 4

# Introduction to Auction Design

In the second part of the thesis, we will study profit-maximization problems related to selling advertisement space on Internet web sites. Clearly, the optimal selling price depends on the benefit derived by the advertisers from the display of their advertisements. In most cases, the web site owner does not have a good estimate of the value of this benefit, as the advertisers' needs often change with time. This is unlike classical optimization problems where the optimizer (the advertisement seller in our case) knows the values of all the variables over which the optimization is to be carried out. In the online advertising market, the web site owners often take recourse to asking the advertisers themselves for their valuations of the benefit they will receive. The value reported by the advertiser is called her *bid*. However, the advertisers, being selfish agents, need not reveal their valuations truthfully if it is in their self-interest to do so. The goal of the web site owner is to set up a selling mechanism that uses these reported values to determine the price(s) to be offered to the advertisers, in such a way that maximizes profit when each agent bids according to her best interest. At the same time, as discussed in Chapter 1, the web site owner would like to keep the process of finding the optimal bidding strategy simple in order to attract more advertisers to the online advertising market. In general, the optimal strategy of an agent depends on how the outcome varies with the bids of various agents as well as on the bids of the other agents. One approach to simplifying the bidding process is to use a dominant strategy mechanism which we discuss in the next section.

## 4.1 Mathematical Framework

*Game Theory* is a branch of mathematics that attempts to understand how rational agents behave in strategic situations. One area of game theory, called *mechanism design*, deals with the construction of mechanisms, or games which are designed to get the agents to behave in a certain way so as to achieve some desired outcome. In this thesis, we adopt the algorithmic mechanism design framework set forth by Nisan and Ronen [NR99]. The mechanism deals with a set of agents $\mathcal{N}$ and wishes to choose from a collection of outcomes $\mathcal{O}$. Each agent $i$ is assumed to have a *valuation function* $v_i : \mathcal{O} \to \mathbb{R}$. We will sometimes refer to the valuation function as the *type* of the agent. Intuitively, the valuation function describes agent $i$'s intrinsic preference for each outcome. An agent's valuation function is known only to that agent. Let $n = |\mathcal{N}|$ be the number of agents and let $\mathcal{T}$ denote the set of all possible valuation functions $\mathcal{O} \to \mathbb{R}$. Let $\mathbf{v} \in \mathcal{T}^n$ denote the vector of types of all agents. The mechanism works by asking each agent to report her type and computing an outcome and a set of payments based on the reported types. We refer to $i$'s reported type as her *bid* $b_i$, and let $\mathbf{b} \in \mathcal{T}^n$ denote the vector of bids. Let $P_i : \mathcal{T}^n \to \mathbb{R}$ be the payment made by agent $i$, and $P = (P_i)_{i \in \mathcal{N}}$ be the payment *scheme*. Thus, a mechanism $\mathcal{M}$ consists of a pair $(O, P)$, where $O : \mathcal{T}^n \to \mathcal{O}$ is the *output* function and $P$ is the payment scheme. A mechanism is deterministic if the output function and the payment scheme are a deterministic function of the bid vector. A mechanism is randomized if the procedure by which the auctioneer computes the output and the payments is randomized.

Each agent's goal is to maximize her *utility* function, which is assumed to be of the form $u_i(O, P_i) = v_i(O) - P_i$. In the literature, this form of utility function is called *quasi-linear*. Since the mechanism determines the outcome and the payments based on the bid vector $\mathbf{b}$, we will often abbreviate $u_i(O(\mathbf{b}), P_i(\mathbf{b}))$ and $v_i(O(\mathbf{b}))$ to $u_i(\mathbf{b})$ and $v_i(\mathbf{b})$ respectively.

Clearly, an agent's utility depends not only on her valuation function, but also on the bid vector. Let $\mathbf{b}_{-i} = (b_1, \ldots, b_{i-1}, ?, b_{i+1}, b_n)$ denote the vector of bids with agent $i$'s bid hidden by a question mark. We will sometimes refer to this as the *masked bid vector*. Similarly, let $\mathbf{v}_{-i}$ denote the vector of all other agents types, and let $\mathcal{T}_{-i} = \mathcal{T}^{n-1}$ be the space of those type vectors. For convenience, we will write $\mathbf{b}$ as $(\mathbf{b}_{-i}, b_i)$. A strategy $S_i : \mathcal{T} \to \mathcal{T}$ is said to be a *dominant strategy* for agent $i$ if $u_i(\mathbf{b}_{-i}, S_i(v_i)) \geq u_i(\mathbf{b}_{-i}, b_i)$

for all $\mathbf{b}_{-i} \in \mathcal{T}_{-i}$ and $b_i, v_i \in \mathcal{T}$; in other words, agent $i$'s best strategy is to report her type as $S_i(v_i)$ whenever her true type is $v_i$. A mechanism is said to be a *dominant strategy mechanism* if every agent has a dominant strategy.

**Definition 4.1 (Truthful Mechanism)** *A truthful mechanism is a dominant strategy mechanism in which truth-telling is a dominant strategy for every agent, i.e.*

$$u_i(\mathbf{b}_{-i}, v_i) \geq u_i(\mathbf{b}_{-i}, b_i) \quad \forall \ \mathbf{b}_{-i} \in \mathcal{T}_{-i} \textit{ and } b_i, v_i \in \mathcal{T}_i.$$

**Theorem 4.1.1 (Bid-independence principle)** *If the mechanism $(O, P)$ is truthful, and $O(\mathbf{b}_{-i}, b_i) = O(\mathbf{b}_{-i}, b_i')$, then $P_i(\mathbf{b}_{-i}, b_i) = P_i(\mathbf{b}_{-i}, b_i')$.*

**Proof:** Suppose the contrary, i.e. $P_i(\mathbf{b}_{-i}, b_i) > P_i(\mathbf{b}_{-i}, b_i')$ while $O(\mathbf{b}_{-i}, b_i) = O(\mathbf{b}_{-i}, b_i')$. When $v_i = b_i$ and all other agents bid $\mathbf{b}_{-i}$, agent $i$ is better off lying and bidding $b_i'$, contradicting truthfulness. □

This principle asserts that in a truthful mechanism, the bid of an agent affects the payment made by the agent only through its effect on the outcome of the mechanism.

## 4.2 Truthfulness as a Means to Simplified Bidding

As mentioned earlier, we would like to construct mechanisms for selling advertising space on Internet web sites under which it is easy for the advertisers to determine their optimal bidding strategies. Unless an advertiser has a dominant strategy, she would be forced to speculate (or hire someone to speculate for her) on how the other advertisers are going to bid in order to determine her optimal strategy. Thus, in order to get rid of speculation and keep the process of bidding simple, we would like each advertiser to have a dominant strategy, i.e. we would like to use a dominant strategy mechanism. Now, we invoke the *Revelation principle* stated below to conclude that we can restrict our attention to truthful mechanisms without missing any possible combination of outcome and payment functions (when we view the outcome and payment as a function of valuation rather than bids).

**Theorem 4.2.1 (Revelation principle [Mye79, DHM79])** *Every dominant strategy mechanism can be converted to a truthful mechanism without changing the outcome or the payments on any type vector (i.e., vector of valuation functions of all agents).*

**Proof:** Given a dominant strategy mechanism $M$, construct a truthful mechanism $M'$ that simulates each bidder's dominant strategy in $M$. Let $S_i$ be agent $i$'s dominant strategy under $M$. Then, $M'$ produces the same outcome and payments on bid vector $\mathbf{b}$ as the mechanism $M$ produces on the bid vector $(S_i(b_i))_{i \in \mathcal{N}}$. Clearly, agent $i$'s dominant strategy under $M'$ is to bid $b_i = v_i$, because this is equivalent to playing the dominant strategy $S_i(v_i)$ in $M$. □

## 4.3 The Vickrey Auction

We next give an example of a classical truthful mechanism due to Vickrey [Vic61]. Given the nature of its payment scheme, it is also called the *second-price auction*. It is used for selling a single item; so the outcome consists of the item being given to one of the agents. The valuation functions of the agents take a very simple form: if an agent gets the item, her valuation of the outcome is $v_i$ (here we are slightly abusing notation by using the term $v_i$ to refer to a single number); otherwise her valuation of the outcome is $0$. The auction consists of inviting bids for the item, and giving the item to the highest bidder and charging her an amount equal to the second-highest bid. If two bidders tie for the highest bid, the item goes to the bidder with the lower index. The proof of the following theorem is simple, but instructive.

**Theorem 4.3.1** *The Vickrey auction is truthful.*

**Proof:** In order to prove truthfulness, we have to show that none of the bidders can benefit by not revealing her true valuation function. Fix an agent $i$ and let $h$ be the highest bid among the other agents. If $v_i > h$, then the agent gets the item for an amount $h$ whenever she bids an amount higher than $h$ (which includes bidding her true valuation function). In this case, her utility is $v_i - h$. The only possible way she can change the outcome is by bidding no more than $h$ in which case her utility is $0$. On the other hand, if $v_i \leq h$, she

makes a profit of $0$ as long as she bids $h$ or less (which includes bidding her true valuation). She can possibly change the outcome by raising her bid above $h$, in which case her utility becomes negative. This shows that bidder $i$ cannot benefit by not bidding her true valuation. □

In fact, it can be shown that bidding one's true valuation is the only dominant strategy under the Vickrey auction. This is so because for every bid value $b_i \neq v_i$, there exists a $\mathbf{b}_{-i}$ for which this bid value is sub-optimal. In particular, bid value $b_i$ is sub-optimal whenever $\max_{j \neq i} b_j$ lies strictly between $b_i$ and $v_i$.

## 4.4 The Vickrey-Clarke-Groves Mechanism

The most celebrated result in truthful mechanism design is the Vickrey-Clarke-Groves (VCG) mechanism [Vic61, Cla71, Gro73] (also see Chapter 23 of [MCWG95]). It is a generalization of the Vickrey auction and can be used when the goal of a mechanism is to maximize the total valuation of all the agents. The VCG mechanism $(O, P)$ is given by:

$$\begin{aligned} O(b) &= o^* \qquad \text{where } o^* \in \text{argmax}_{o \in \mathcal{O}} \sum_{i \in \mathcal{N}} b_i(o) \\ P_i(b) &= -\textstyle\sum_{j \neq i} b_j(o^*) + h_i(\mathbf{b}_{-i}) \end{aligned}$$

where the functions $h_i$ are arbitrary. That is, VCG selects the outcome that maximizes the total reported valuation, and charges agent $i$ an amount $P_i$ that depends on $b_i$ only through its influence on the outcome $o^*$, just as required by bid-independence principle. Since the $h_i$ terms in the payment are completely independent of agent $i$'s bid, they are irrelevant to truthfulness. The $\sum_{j \neq i} b_j(o)$ term in the payment is quite special though — it aligns the utility function of agent $i$ with the utilitarian objective function. This makes the mechanism truthful, as asserted by the following theorem.

**Theorem 4.4.1** *The VCG mechanism is truthful.*

The basic VCG mechanism can be augmented by weighting the agents differently and adding a bias to the outcome function, while preserving truthfulness [GL77, Rob79]. More

formally, let $w \in \mathbb{R}_+^{\mathcal{N}}$ be a set of non-negative weights. Let $H : \mathcal{O} \to \mathbb{R}$ be a "bias" function. The resulting *weighted, biased VCG mechanism* is defined by:

$$\begin{aligned} O(b) &= o^* && \text{where } o^* \in \text{argmax}_{o \in \mathcal{O}} \left( \sum_{i \in \mathcal{N}} w_i b_i(o) + H(o) \right) \\ P_i(b) &= -\tfrac{1}{w_i} \left( \textstyle\sum_{j \neq i} w_j b_j(o^*) \right) + h_i(\mathbf{b}_{-i}) && \text{when } w_i > 0 \\ P_i(b) &= h_i(b_i) && \text{when } w_i = 0 \end{aligned}$$

where the functions $h_i$ are arbitrary.

**Theorem 4.4.2** *For every choice of weights and bias function, the weighted, biased VCG mechanism is truthful.*

## 4.5 Analysis Framework

Our goal is to design truthful auctions that maximize the auctioneer's profit. Ideally, we would like to design optimal auctions — auctions that, on any given input, perform better (or at least no worse) than any other truthful auction. In fact, we do precisely that for the problem of selling multiple advertisement slots on a single web page when there are no budget constraints and the outcome function comes from a certain natural class of functions. If we are unable to provide an auction that is optimal for all inputs, our next goal would be to design auctions that perform well compared to a meaningful benchmark. In this case, we will analyze them in the competitive analysis framework of Goldberg et al. [GHW01] for profit-maximizing auctions. Prior to [GHW01], profit maximization in mechanism design was considered in a Bayesian framework. In such a framework, it is assumed that the agents' valuations are drawn from some probability distribution and that the mechanism designer has knowledge of this prior distribution. The goal is to design the Bayesian optimal mechanism for the given prior distribution. The obvious drawback of this approach is that the mechanism designer must know the prior distribution, which is exactly the pre-requisite that we are trying to eliminate. Therefore, we would like to design a truthful auction mechanism that performs well without knowing anything about the input,

i.e., a single mechanism that works well for all inputs and does not need to have any prior knowledge about the preferences of various agents.

We will look to design an auction that obtains a profit that is comparable to the profit of some natural benchmark. The competitive ratio of an auction will be defined as the supremum over all possible input bid vectors, of the ratio of the benchmark revenue to the revenue of the auction. Let $r_{\mathcal{A}}(\mathbf{b})$ be the revenue of an auction $\mathcal{A}$ on the bid vector $\mathbf{b}$ and let $r_{OPT}(\mathbf{b})$ be the benchmark revenue on $\mathbf{b}$. Then the competitive ratio of auction $\mathcal{A}$ is defined as:

$$\sup_{\mathbf{b}} \frac{r_{OPT}(\mathbf{b})}{r_{\mathcal{A}}(\mathbf{b})}$$

A key part of setting up a competitive framework for analyzing solutions to a problem is coming up with the right metric for comparison. As a starting point, we could try competing with the strongest possible benchmark, namely the profit of an auctioneer who is *omniscient*, i.e., knows the valuations of all the agents, and can use them to compute the outcome and the payments. However, in most interesting auction design problems, it can be shown that it is not possible for a truthful auction to achieve any finite competitive ratio against such a powerful benchmark. The next step is to constrain the benchmark auctioneer in some reasonable manner. Consider the problem of selling multiple identical units of an item to a set of indistinguishable agents each of whom desires one unit of the item. In this case, we can specify the outcome of the auction and the payments by specifying an offer price for each agent — an agent gets a unit if and only if her bid is no less than the price offered to her. If the same price $p$ is offered to all the agents, the revenue is $p \cdot n(p)$, where $n(p)$ denotes the number of agents whose valuation (and bid in a truthful auction) is at least $p$. One possible benchmark is the revenue earned by the ex post (or omniscient) optimal single price that can be offered to all the agents. We denote this revenue by $\mathcal{F}$ or $\mathcal{F}(\mathbf{b})$.

**Theorem 4.5.1 ([GHK$^+$02])** *For any truthful auction $\mathcal{A}$, and any $\beta \geq 1$, there is a bid vector $\mathbf{b}$ such that the expected profit of $\mathcal{A}$ on $\mathbf{b}$ is less than $\mathcal{F}(\mathbf{b})/\beta$.*

The intuition behind this result is that if one agent has a bid much higher than the rest, then offering the optimal single price will result in a revenue equal to this bid. However, in the absence of competition, there is no truthful way to extract this high bid value from the agent. This benchmark can be generalized as follows.

**Definition 4.2** *Let $P_i$ be the set of all prices $p$ such that at least $i$ agents have bids $p$ or higher. Then $\mathcal{F}^{(i)}$ denotes the maximum possible revenue from offering the same price $p \in P_i$ to all the agents.*

Goldberg et al. [GHW01] use $\mathcal{F}^{(2)}$ as the benchmark for their competitive analysis. (Note that $\mathcal{F}^{(2)}$ is identical to $\mathcal{F}$ whenever the optimal single price can be afforded by at least two agents.) This benchmark seems reasonable when the agents are indistinguishable and [GHK$^+$02] shows that indeed no truthful mechanism in a large natural class of auction mechanisms can outperform this benchmark. To make this more precise, let an auction mechanism be called *monotonic* if an agent's chances of getting an item do not decrease as her bid increases. Then, [GHK$^+$02] prove the following theorem.

**Theorem 4.5.2** *For any set of bids $\mathbf{b}$, the revenue of a monotonic, truthful auction is no more than $\mathcal{F}(\mathbf{b})$.*

We note here that the auctions that compete with a given benchmark need not satisfy the constraints that the benchmark pricing function has to meet. For example, for the above problem, we do not require the auction to offer the same price to all the agents. In fact, Goldberg and Hartline [GH03b] show the following.

**Theorem 4.5.3** *Let $\mathcal{F}_m$, denote the revenue of the optimal single price at which at least $m$ of the agents get a unit. If a truthful auction always offers the same price to all bidders, it cannot be $O(\log \frac{n}{m} / \log\log \frac{n}{m})$-competitive with $\mathcal{F}_m$.*

Next, consider the *attribute auction problem*, or the problem of selling identical copies of an item to agents who can be distinguished on the basis of some known attribute, like the postal zip code of the agent. In this case, the benchmark auctioneer can be allowed to divide the agents into, say, $r$ sets (based on their attributes), and offer different prices to different sets (but the same price to all agents in the same set). Blum and Hartline [BH05] use the optimal revenue from such a pricing function as the benchmark. When we are trying to sell a divisible good, e.g. network bandwidth, and the attribute represents the demand of the agent, e.g. the amount of bandwidth the agent needs, it seems reasonable to constrain the auctioneer to offer higher prices (or at least not lower prices) to agents with higher

attributes. We call such a pricing function *monotone* as the offer prices are monotone in attribute, and we will use the profit of the optimal such pricing function as the benchmark for analyzing the auctions presented in Chapter 5.

### 4.5.1 Some Competitive Auction Results

The problem of designing a truthful auction for selling multiple identical units of an item that performs well in the worst-case has received considerable attention recently [GHW01, FGHK02, GH03a, GHKS04, HM05]. The current best truthful auction [HM05] achieves a competitive ratio of $3.25$ with respect to the revenue of the optimal single price that sells to at least two agents.

The attribute auction problem was introduced by Blum et al. [BH05] who investigated whether it was possible to get a revenue higher than the revenue of the optimal single price when the auctioneer is able to distinguish between agents based on a known attribute of the agents. Let $OPT(r)$ be the profit of an omniscient auctioneer who is allowed to divide the agents into $r$ sets (based on their attributes), and offer different prices to different sets, and let $h$ be an upper bound on the value of the highest bid. Then, [BH05] presents a truthful auction that achieves a revenue of at least $\mathrm{OPT}(r)/16 - rh/2$ simultaneously for all $r$.

# Chapter 5

# Auctions for a Single Advertisement Slot: The Knapsack Auction Problem

Consider a web page with a single slot where an advertisement can be displayed. Whenever an Internet user accesses the web page, the web site owner can choose to show her an advertisement. The process of displaying an advertisement is called an *impression*. Depending on the content of the web page, a variety of advertisers might be interested in displaying advertisements on the web page. Each of these advertisers will invoke a different level of interest from the Internet users visiting the web page. Thus, each advertiser will have a different click-through rate (CTR) associated with her advertisement (the click-through rate of an advertisement is the fraction of its impressions that result in a click by an Internet user). We assume that the web site owner has (or can collect) statistical information about the CTRs of various advertisers. Moreover, each advertiser might have a limit on the number of clicks on her advertisement that she can handle in a single day. We assume that this limit is known to the web site owner. One scenario in which this assumption is justified is when the web site owner interacts with the advertisers repeatedly. Thus, the web site owner knows the maximum number of impressions $c_i$ desired by an advertiser $i$, which can be computed as the product of the maximum number of clicks desired and the average number of impressions needed to generate a click (the reciprocal of the CTR). This is called the *demand* (or *size*) of the advertiser. In addition, the web site owner has an estimate of the total number of impressions available during the course of the day. He would

like to allocate the available impressions among the interested advertisers with the goal of maximizing profit. To make an informed decision, he needs to know the benefit that each advertiser derives from a click on her advertisement. However, an advertiser's valuation of a click or an impression is known only to the advertiser. We let $v_i$ denote advertiser $i$'s valuation of $c_i$ impressions. The fact that the valuations are private implies that the web site owner has to resort to soliciting bids from the advertisers and running an auction to sell the advertisement space. As discussed in the previous chapter, unless the seller uses a dominant strategy mechanism, the optimal bidding strategy of an advertiser will depend on the bids of other agents, making the process of bidding complex. Also, by the revelation principle, every dominant strategy mechanism has an equivalent truthful mechanism. Thus, motivated by the desire to keep bidding simple, we would like to design a *truthful* mechanism for selling advertisement space with the goal of maximizing profit.

**Outline of the Chapter.** The rest of the chapter is organized as follows. We define the problem formally in Section 5.1. We discuss the analysis framework and state our results in Section 5.2. Related work is discussed Section 5.3. In Section 5.4, we present a comparison of the different pricing rules that we consider in this chapter. In Section 5.5, we discuss the algorithmic complexity of computing the optimal pricing function without any consideration of the game-theoretic issues. We present competitive auctions for our problem in Section 5.6. For this, we first show (in Section 5.6.1) how to reduce the limited-supply auction problem to the unlimited-supply auction problem with a small loss in approximation factor. Then in Section 5.6.2, we give an unlimited-supply auction that achieves a constant fraction of the benchmark revenue (with a small additive loss).

## 5.1 Problem Definition

We model our problem as the following abstract *private-value* version of the (fractional) *knapsack problem*. We will refer to it as the *knapsack auction problem*. There is a set of $n$ agents $\mathcal{N} = \{1, \ldots, n\}$, each of whom has an object. Let $c_i$ represent the publicly-known size of agent $i$'s object. Each of these agents desires to have her object placed in a knapsack with total capacity $C$. Our goal is to design a single-round, sealed-bid auction for

this setting. In this auction, an outcome is characterized by a set of agents called *winners*. The winning agents have their items placed into the knapsack and losing agents do not (although our auction will only accept whole items, it will be shown to perform well with respect to benchmarks that are allowed to accept fractional items). A set of agents forms a feasible outcome if and only if the total demand of all the winning agents does not exceed the capacity of the knapsack. Let $v_i$ denote agent $i$'s *valuation* for having their item placed in the knapsack. This valuation represents the benefit received by the agent from winning. (Note that we are again abusing notation by using $v_i$ to refer to a single number rather than a function as in Section 4.1). We can also define a fractional version of the knapsack auction problem, where the auctioneer can accept a fraction of each object and bidder $i$ values the acceptance of $\phi_i$ fraction of its object at $\phi_i v_i$. We assume that all the agents attempt to maximize their *utility*, measured as the difference between their valuation and their payment.

We will assume that the all the agents' valuations fall within a known range $[1, h]$. We denote by $\mathbf{b} = (b_1, \ldots, b_n)$ the vector of bids submitted by the agents and by $\mathbf{c} = (c_1, \ldots, c_n)$ the vector of publicly-known object sizes. We assume for convenience that the agents are indexed by decreasing size, i.e., $c_1 \geq c_2 \geq \cdots \geq c_n$. Following [BH05], we sometimes refer to these object sizes as *attributes*. An auction's *profit*, $\mathcal{A}(\mathbf{b}, \mathbf{c})$, is the sum of the payments of the winning agents.

Our goal is to design a profit-maximizing truthful mechanism. Recall that in a truthful mechanism, each advertiser's optimal strategy, irrespective of others' bids, is to bid her true valuation. In auction problems like the current one, truthful mechanisms are known to have the following algorithmic characterization [GHW01]. Related formulations to the one below have appeared in numerous places in recent literature, e.g., [AT01, FGHK02, LOS99]. Recall that the masked bid vector $\mathbf{b}_{-i}$ denotes the vector $\mathbf{b}$ with $b_i$ replaced with a '?', i.e., $\mathbf{b}_{-i} = (b_1, \ldots, b_{i-1}, ?, b_{i+1}, \ldots, b_n)$.

**Definition 5.1 (Bid-Independent Auction)** *Let $f$ be a function that maps any masked bid vector (with a '?') and attribute vector pair to a price (non-negative real number). The* deterministic bid-independent auction defined by $f$, $\mathrm{BI}_f$, *works as follows. For each agent $i$:*

1. *Set* $t_i = f(\mathbf{b}_{-i}, \mathbf{c})$.
2. *If* $t_i < b_i$*, agent* $i$ *wins at price* $t_i$.
3. *If* $t_i > b_i$*, agent* $i$ *loses.*
4. *Otherwise (*$t_i = b_i$*), the auction can either accept the bid at price* $t_i$ *or reject it.*

*A randomized bid-independent auction is a distribution over deterministic bid-independent auctions.*

The proof of the following theorem can be found, for example, in [FGHK02].

**Theorem 5.1.1** *An auction is truthful if and only if it is equivalent to a bid-independent auction.*

**Prior-free Optimization.** Through the study of the knapsack auction problem, we wish to develop a better understanding of how to do prior-free optimization (i.e. optimization without assuming any prior knowledge of the distribution of the valuations) when there are non-trivial constraints on the allocation. In our case, items selected for the knapsack must all fit in the available space. A similar direction was attempted by Fiat et al. in [FGHK02] for the *multicast pricing* problem in which an obvious market segmentation could be exploited to reduce the problem from a *private-value* optimization problem to a *public-value* optimization problem. In the knapsack auction problem, however, there is no obvious market segmentation and indeed, figuring out how to segment the agents into markets in a truthful manner constitutes a key portion of the solution. To the best of our knowledge, this work represents the first solution to a non-trivial private-value optimization problem when market segments are not given in advance.

## 5.2 Analysis Framework and Results

As discussed in Chapter 4, we analyze the performance of our auctions in the *competitive analysis* framework of Goldberg et al. [GHW01] for profit-maximizing auctions. We are looking to design an auction that obtains a profit that is comparable to the profit of a natural

optimal *omniscient* auction. As in the bid-independent auction (see Definition 5.1), we will specify the allocation and payments through a pricing function that maps agents to offer prices — each agents with a bid above her offer price gets her item placed in the knapsack, while the items of all the agents with bids below their offer price get rejected. For any agent whose bid is equal to her offer price, the auctioneer has the discretion to either accept her item or reject it.

For the case that the agents are indistinguishable, [GHW01] uses the profit of the ex post optimal single price that can be offered to all agents as the benchmark. For this benchmark, the offer price can not depend on the agent at all and the pricing function is constant. We refer to such pricing schemes as *uniform pricing*. While this benchmark is reasonable for indistinguishable agents (see Theorem 4.5.2), it might be too weak when the seller knows the value of some attribute of the agents (their demand in our case) which can be used as a basis to distinguish between them. In this case, it is more reasonable to let the pricing function vary with the attribute value (but remain constant for any given attribute value). One natural candidate pricing strategy is to charge the same price per unit demand. Thus, agents demanding more of the capacity pay proportionally more. We refer to such pricing schemes as *linear pricing*, since the only valid pricing functions are linear in demand. The most general pricing strategy we will consider follows the least restrictive natural assumption we can place on prices: that agents desiring more capacity not pay less than those desiring less. We refer to this as *monotone pricing*, since the only valid pricing functions for this class are monotone non-decreasing in demand. Clearly, both uniform pricing and linear pricing are special cases of monotone pricing. As such, we will use the revenue of the optimal omniscient monotone pricing scheme as the benchmark revenue throughout the chapter. We note that an auction that achieves a constant fraction of the revenue of the optimal monotone pricing function also achieves a constant fraction of the revenue of optimal uniform and linear pricing functions.

We would like to note here that our auctions will not produce a monotone pricing function — in fact, it can be shown that any truthful auction that always produces a monotone pricing function cannot perform well with respect to any of the above benchmarks. To see this, recall Theorem 4.5.3 which shows that for the case of indistinguishable agents, no truthful auction that offers the same price to all the agents can perform well with respect to

the revenue of the ex post optimal single price for all agents. Next, note that for the special case of our problem where all the agents have the same attribute value, producing a monotone pricing function is equivalent to offering the same price to all the agents; in addition, all the above benchmarks reduce to the revenue of the ex post optimal single price. Thus, we get the following corollary of Theorem 4.5.3.

**Corollary 5.2.1** *Let $\mathcal{F}_m$, denote the revenue of the optimal single price at which at least $m$ of the agents get a unit. If a truthful auction always produces a monotone pricing function, it cannot be $O(\log \frac{n}{m} / \log\log \frac{n}{m})$-competitive with $\mathcal{F}_m$.*

## Our Contributions

Let $\mathrm{OPT}$ be the revenue obtained by the best monotone pricing function. We can apply Theorem 4.5.1 to the case when all the agents have the same attribute value to conclude that it is not possible to obtain a constant fraction of $\mathrm{OPT}$ in the worst case. We design auctions that obtain at least a constant fraction of $\mathrm{OPT}$ minus a small additive loss term, i.e., $\alpha\,\mathrm{OPT} - \lambda h$ (where $h$ is an upper bound on the highest bidder's valuation). Ideally both $\alpha$ and $\lambda$ should be constants. Here, we achieve a constant $\alpha$ and $\lambda \in O(\log\log\log n)$.

We first consider a special case of the knapsack auction problem, namely the *unlimited-supply* case where the capacity of the knapsack, $C$, exceeds the total demand, $\sum_i c_i$. This is an interesting special case of the original problem that is much less constrained. We present a truthful auction that achieves a revenue of $\alpha\,\mathrm{OPT} - O(\log\log\log n)h$. We then use this auction to solve the limited-supply case. In doing so, we outline a general approach for dealing with non-trivial optimization problems. The first step of this approach is to solve the *unlimited-supply* version of the problem. The second step is to reduce the *limited-supply* (or general) version of the problem to the unlimited-supply version. This approach works in general for "monotone" optimization problems, where the feasibility of an allocation implies the feasibility of all subsets of the allocation as well. The reduction works by

(a) selecting a set of agents that can all be allocated together,

(b) running the unlimited-supply solution on this selected set, and

(c) deciding offer prices based on the prices determined in step (b) and some other criteria discovered in step (a). Step (a) and (c) must be done rather carefully to preserve incentive-compatibility and the competitive ratio.

In order to preserve a constant competitive ratio, the optimal unlimited-supply monotone pricing on the selected agents must be within a constant factor of the optimal limited-supply monotone pricing on the original set of agents. To preserve incentive compatibility, step (a) and (c) must be performed to meet additional conditions discussed later.

## 5.3 Related Work

The private-value knapsack problem has earlier been studied by Mu'alam et al. [MN02] with the objective of maximizing social welfare, rather than profit. The problem of designing profit-maximizing auctions for selling advertisements on Internet web sites has been the subject of some recent work [MSVV05, BCI$^+$05]. Mehta et al. [MSVV05] ignore the game-theoretic issues and instead focus on the algorithmic problem of matching advertisers to web pages when their valuations and budgets are known to the auctioneer. Borgs et al. [BCI$^+$05] study the problem of selling multiple identical units when the agents are interested in getting multiple units as long as their payment does not exceed their budget. Both the valuation and the budget of an advertiser are considered private values. As such, unlike our setting, they are unable to distinguish between bidders and are only able to compete with the revenue of the optimal single price at which at least two agents get a unit.

A problem closely related to ours is the *attribute auction problem*. It was introduced by Blum et al. [BH05], who investigated whether it was possible to get a higher profit when the auctioneer is able to distinguish between agents based on a known attribute of the agents. They presented a solution for the unlimited-supply, single-dimensional-attribute auction problem, and analyzed its performance by comparing its profit to that of the optimal piecewise-constant (not necessarily monotone) pricing rule. We will henceforth refer to this attribute auction as the *general attribute auction*. We will be make use of the following result.

**Theorem 5.3.1** *[BH05] Let $OPT(r)$ be the profit of an omniscient auctioneer who is*

*allowed to divide the agents into $r$ sets (based on their attributes), and offer different prices to different sets. Also, let $h$ be an upper bound on the value of the highest bid. Then, the* general attribute auction *obtains a profit of at least $\mathrm{OPT}(r)/16 - rh/2$ simultaneously for all $r$.*

## 5.4 Pricing Rules

For a given instance of the knapsack problem, we consider a pricing rule and selection of items to be contained in the knapsack to be *valid* if

1. the valuation of each item in the knapsack is at least equal to the price set for items of that size, and
2. the valuation of each item not in the knapsack is no more than the price set for items of that size.

The *payoff* of a particular pricing rule is simply the number of items in the knapsack at each size times the price for items of that size summed over all item sizes. The goal of a pricing algorithm, then, is to find a valid pricing rule that maximizes the total payoff. We note that the validity conditions can be viewed as a requirement for the pricing rule and assignment to be *envy-free* [GHK$^+$05] in the sense that each agent prefers her outcome to the outcome of every other agent, or equivalently that none of the agents is envious of the outcome of another agent.

As mentioned in the introduction, we will be primarily interested in three classes of pricing functions: *uniform pricing*, *linear pricing* and *monotone pricing*. In uniform pricing, we will assume that there is a single price for all item sizes. In linear pricing, we will assume that there is a single price per unit size. Finally, our most general class of pricing functions will be monotone pricing where the price has to be a non-decreasing function of the item sizes. One can view the restriction to monotone prices as an additional requirement for envy-freedom, since without monotone prices, a small object would be envious of a larger object being placed into the knapsack at a smaller price than it.

We now consider the worst case relationship between optimal uniform pricing, optimal linear pricing, and optimal monotone pricing. As uniform and linear pricing are a special

case of monotone pricing, it is clear that the profit of the optimal monotone pricing is better than that of both the optimal uniform and the optimal linear pricing. We now get bounds on how much worse uniform and linear pricing can be.

**Lemma 5.4.1** *Uniform pricing can be a logarithmic factor worse than monotone and linear pricing and this is tight.*

**Proof:** Consider $n$ items with $v_i = c_i = 1/i$ and $C = \infty$. Optimal monotone and linear pricing use the pricing rule $\pi(c) = c$ for a total payoff of $\sum_i 1/i = \Theta(\log n)$. Uniform pricing on the other hand must choose a single price $\pi(c) = p$. Since the number of items with value at least $p$ is at most $1/p$, the total payoff of uniform pricing is at most 1. This provides the desired logarithmic factor separation. Tightness follows from the following observation: for any $v_1, \ldots, v_n$ reordered such that $v_i \leq v_{i+1}$, the payoff of the optimal uniform price is given by $\max_i iv_i$. Since $v_i \leq \frac{\max_i iv_i}{i}$ for all $i$, the maximum possible payoff $\sum v_i \leq \log n \cdot \max_i iv_i$. □

**Lemma 5.4.2** *Linear pricing can be a linear factor worse than monotone and uniform pricing and this is tight.*

**Proof:** Take $C = \infty$, $v_i = 1$, and $c_i = n^{-(i-1)}$ for all $i$. Optimal monotone and uniform pricing set $\pi(c) = 1$ and obtain a payoff of $n$. The optimal linear pricing function uses the pricing function $\pi(c) = c$. This gives a payoff of $\sum_i n^{-(i-1)} = O(1)$. To see tightness, note that linear pricing can always obtain a payoff of at least $\max_i v_i$, which is at least $(1/n)^{th}$ of the optimal monotone and uniform pricing payoffs. □

These results imply that asymptotically, monotone pricing is more powerful than uniform or linear pricing.

## 5.5 Pricing Algorithms

In this section, we explore the non-game-theoretic problem of designing good *knapsack pricing algorithms*. As in the mechanism design problem, we assume that the knapsack has size $C$ and that item $i$ has size $c_i$ and value $v_i$. These knapsack pricing algorithms differ

from conventional knapsack algorithms in that the payoff earned by the seller from placing an item in the knapsack is not the item's value, but instead a price that is a function of the size of the item.

First note that if we were allowed to accept objects fractionally, then a greedy pricing strategy would yield an optimal algorithm for each of the three pricing policies. However, when the objects are indivisible, we show the following lemmas.

**Lemma 5.5.1** *Optimal uniform pricing is in P.*

**Lemma 5.5.2** *Optimal linear pricing is NP-hard.*

**Lemma 5.5.3** *Optimal monotone pricing is NP-hard.*

Lemma 5.5.1 follows from the following simple procedure for computing the profit for any offer price $p$. First add all items with value strictly greater than $p$; these must be in any knapsack when price $p$ is offered. If this exceeds the capacity of the knapsack, then $p$ is an infeasible offer price. Otherwise, add the items with value equal to $p$ to the knapsack from smallest to largest. This maximizes the number of items in the knapsack given the offer price of $p$. Given this procedure, we can find the optimal uniform offer price by searching through the $n$ item values, $v_1, \ldots, v_n$, as possible offer prices.

Lemmas 5.5.2 and 5.5.3 follow from the hardness of the subset-sum problem by the following simple reduction. Given an instance of the subset-sum problem with item $i$ having size $\hat{c}_i$, create an instance of the knapsack problem with the same number of items, and set $v_i = c_i = \hat{c}_i$ for all $i$. Set the knapsack capacity to $S$, the desired subset sum. The optimal pricing function is simply $\pi(x) = x$ (which is linear and therefore monotone); however, the algorithm still has the discretion to "break ties" by choosing which subset of the items to put in the knapsack (the validity conditions are satisfied for any subset of the items). The reduction is complete when we observe that there exists a subset of items with sum $S$ if and only if the optimal profit for this knapsack problem instance is $S$.

### 5.5.1 Pricing Algorithms for Unlimited Supply

An interesting special case of the knapsack pricing problem, referred to as the *unlimited-supply* problem, is the case where $C$ is effectively infinite, i.e. $C \geq \sum_i c_i$. We first note that

with unlimited supply, an optimal algorithm would either accept an object in its entirety or reject it completely. Thus, the optimal fractional pricing rule is identical to the optimal integral pricing rule. It turns out that the unlimited-supply cases of the knapsack pricing problem are relatively easy to solve in polynomial time. Contrast this with the envy-free pricing problems from [GHK$^+$05], where even simple special cases of the unlimited-supply pricing problems considered are APX-hard, i.e., unless $P = NP$, there is no polynomial time approximation scheme (PTAS).

**Lemma 5.5.4** *Optimal unlimited-supply linear pricing is in P.*

**Proof:** To compute the optimal linear pricing for items $1, \ldots, n$, item $i$ with value $v_i$ and size $c_i$, we compute each item's value per unit size, $d_i = v_i/c_i$. For each price rate $d_i$, the payoff of the algorithm is $d_i \times \sum_{j\,:\,d_j \geq d_i} c_j$. Payoffs for all values of $d_i$ can easily be computed in $O(n \log n)$ time by first sorting the items by $d_i$. □

**Lemma 5.5.5** *Optimal unlimited-supply monotone pricing is in P.*

**Proof:** The optimal monotone pricing rule can be computed by using the following dynamic programming algorithm. Intuitively, the table entry $T[i, p]$ corresponds to the optimal payoff from the smallest $i$ items using monotone prices, when the highest price offered is $p$.

1. Order the items by increasing size, i.e. $c_1 \leq c_2 \cdots \leq c_n$.

2. Solve the following dynamic program for all $p \in \{v_1, \ldots, v_n\}$.

$$T[0, p] = 0$$
$$T[i, p] = \text{profit}(v_i, p) + \max_{q \in \{v_1, \ldots, v_n\};\ q \leq p} T[i-1, q]$$

   with $\text{profit}(v_i, p) = p$ if $v_i \geq p$ and 0 otherwise.

3. Output $\max_{p \in \{v_1, \ldots, v_n\}} T[n, p]$.

□

### 5.5.2 Limited-Supply Approximation via Reduction to Unlimited Supply

We now show how to approximate the optimal monotone knapsack pricing in the general case by using an optimal or approximate pricing algorithm for the unlimited-supply case. A similar technique can be applied to the linear pricing variant of the problem. We consider the following technique for composing two pricing algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$:

**Definition 5.2 (Pricing Algorithm Composition)** *Given two pricing algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, we define the composite algorithm $\mathcal{A}_1 \circ \mathcal{A}_2$ as:*

1. *Run $\mathcal{A}_1$ to obtain pricing function $\pi_1(\cdot)$ and let $H$ be the set of winners.*
2. *Run $\mathcal{A}_2$ on $H$ to obtain pricing function $\pi_2(\cdot)$.*
3. *Output $\pi(x) = \max(\pi_1(x), \pi_2(x))$ for the agents in $H$. For an agent not in $H$, let $\hat{x}$ be the smallest-sized agent in $H$ whose size is no smaller than $x$. Offer $x$ a price of $\pi(x) = \pi(\hat{x})$.*

If algorithm $\mathcal{A}_1$ produces a set of winners $H$ that is feasible (for our knapsack problem, feasibility means that all items in $H$ can fit in the knapsack simultaneously), then we can choose $\mathcal{A}_2$ as the optimal unlimited-supply monotone pricing algorithm. Since the feasible solutions to the knapsack problem are *closed under inclusion*, meaning that any subset of a feasible set is also feasible, $\mathcal{A}_2$ will always produce a feasible set. All we need to argue then is that the composite pricing algorithm yields a monotone pricing and that it performs well.

**Lemma 5.5.6** *If $\mathcal{A}_1$ and $\mathcal{A}_2$ are monotone pricing algorithms, then $\mathcal{A}_1 \circ \mathcal{A}_2$ also yields monotone pricing.*

Let $\mathrm{Winners}_{\mathcal{A}}(X)$ denote the winners of algorithm $\mathcal{A}$ applied to the agent set $X$.

**Definition 5.3 (Performance Preservation)** *Consider a performance benchmark $OPT$, with $OPT(Y)$ denoting the value of the benchmark on the agent set $Y$. An algorithm*

$\mathcal{A}$ approximately preserves *the performance benchmark,* $\mathrm{OPT}$*, if for all inputs with agent set* $X$*,*

$$\mathrm{OPT}(\mathrm{Winners}_{\mathcal{A}}(X)) \geq \alpha\,\mathrm{OPT}(X) - \gamma h$$

*for some constants* $\alpha$ *and* $\gamma$*, where* $h$ *is the maximum value of any item.*

Note that if algorithm $\mathcal{A}_1$ approximately preserves the performance of optimal monotone pricing, and $\mathcal{A}_2$ is optimal (or constant-competitive), then their composition achieves a payoff of at least $\alpha'\,\mathrm{OPT}(\mathcal{N}) - \gamma' h$ for some constants $\alpha'$ and $\gamma'$. We next discuss monotone pricing algorithms that approximately preserve the performance of the optimal monotone pricing and produce a feasible winner set. Recall the standard weighted knapsack problem: given item values $v_1, \ldots, v_n$, item sizes $c_1, \ldots, c_n$, and knapsack capacity $C$, find the set of items, $H$, with maximum total value that simultaneously fit in the knapsack. We present a pricing algorithm based on a natural approximation algorithm for the knapsack problem.[1]

**Algorithm** APPROX-KNAPSACK
**Input:** Items with values $v_1, \ldots, v_n$ and sizes $c_1, \ldots, c_n$ and a knapsack of size $C$.
**Output:** A monotone pricing function $\pi$.

1. Ignore large items with $c_i > C/2$.
2. List the remaining items in the order of decreasing value-per-unit-size, $d_i = v_i/c_i$.
3. Select the largest prefix of the item list that fits in the knapsack as the winner set $H$.
4. Offer $\pi(x) = d^* x$, where $d^*$ is the largest value-per-unit-size among the losers, for $x \leq C/2$ and offer $\infty$ otherwise.

**Lemma 5.5.7** *Algorithm* APPROX-KNAPSACK *approximately preserves the optimal fractional monotone pricing,* $\mathrm{OPT}$*, with* $\mathrm{OPT}(H) \geq \mathrm{OPT}(\mathcal{N})/3 - h$*.*

**Proof:** Let $\mathcal{N}' \subset \mathcal{N}$ be the items with size at most $C/2$. At most one item from the set $\mathcal{N} - \mathcal{N}'$ can fit in the knapsack. Thus, the algorithm can restrict its attention to the set $\mathcal{N}'$ without losing more than an additive term of $h$.

[1] We add Step 4 to output a monotone pricing in addition to a set $H$ that approximates the optimal knapsack.

If all of $\mathcal{N}'$ fits into the knapsack then the theorem follows. Otherwise, the items in the winner set $H$ fill at least half of the knapsack. This is because there is some item in $\mathcal{N}'$ that could not fit into the remaining space of the knapsack, and the items in $\mathcal{N}'$ have size at most $C/2$. Therefore, $\pi(x)$ is a monotone (in fact, a linear) pricing rule that obtains a payoff of at least $d^*C/2$; this is because the value-per-unit-size of agents in $H$ is at least $d^*$. Thus, $\mathrm{OPT}(H) \geq d^*C/2$.

Let $L = \mathcal{N}' \setminus H$ be the items not included in the knapsack (which all have value-per-unit-size at most $d^*$). Clearly, $\mathrm{OPT}(L) \leq d^*C \leq 2\,\mathrm{OPT}(H)$. Therefore, $3\,\mathrm{OPT}(H) \geq \mathrm{OPT}(H) + \mathrm{OPT}(L) \geq \mathrm{OPT}(\mathcal{N}')$. □

Note that the lemma above holds for the optimal *fractional* monotone pricing. As noted before, for unlimited supply, the optimal fractional pricing rule is identical to the optimal integral pricing rule. Thus, we get the following theorem.

**Theorem 5.5.8** *The composite algorithm obtained from Algorithm* APPROX-KNAPSACK *and the optimal unlimited-supply monotone pricing algorithm achieves a payoff of at least* $\mathrm{OPT}/3 - h$*, where* $OPT$ *is payoff of the optimal fractional monotone pricing rule.*

We can similarly define a composition method for linear pricing. The only change that we need to make is in Step 3 of the composition method (see Definition 5.2). Let the two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ output a linear pricing rule. Let the pricing rule output by $\mathcal{A}_1$ be $\pi_1(x) = \tau_1 x$ and the pricing rule output by $\mathcal{A}_2$ be $\pi_2(x) = \tau_2 x$. Then $\mathcal{A}_1 \circ \mathcal{A}_2$ uses the pricing rule $\pi(x) = \max\{\tau_1, \tau_2, \tau\}x$, where $\tau$ is the maximum value-to-size ratio among items with size greater than $C/2$. A theorem similar to Theorem 5.5.8 above holds for this composition method.

## 5.6 Approximately Optimal Knapsack Auctions

In this section, we extend the technique of composing pricing algorithms to mechanism design problems. These techniques suggest a general procedure for reducing limited-supply (or, constrained) problems to unlimited-supply (or, unconstrained) mechanism design problems.

### 5.6.1 Reduction via Composition

Consider any constrained profit maximization problem in a private-value setting, e.g., the single-parameter agent settings of [FGHK02, GH05]. One can think of the unlimited-supply case as that where all outcomes are feasible; whereas the limited-supply case is constrained to produce some outcome in a restricted feasible set. In the case where the set of feasible outcomes (sets of agents) is *closed under inclusion*, meaning that all subsets of a feasible set are also feasible, the following general approach can be attempted: first find a good feasible set, then run an unlimited-supply auction on it. Below we formalize the game-theoretic issues that arise with this approach.

**Definition 5.4 (Mechanism Composition)** *Given two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$, we define the composite mechanism $\mathcal{M}_1 \circ \mathcal{M}_2$ as:*

1. *Simulate $\mathcal{M}_1$ and let $H$ be the set of winners.*
2. *Simulate $\mathcal{M}_2$ on the set $H$.*
3. *For each agent in $H$, offer a price equal to the maximum of the prices offered to her by $\mathcal{M}_1$ and $\mathcal{M}_2$. An agent not in $H$ is declared a loser.*

We will be looking to use this composition technique with a mechanism $\mathcal{M}_1$ that always outputs a set of winners for which all subsets are feasible, and a mechanism $\mathcal{M}_2$ which takes such a set of agents (i.e., a set with respect to which the mechanism effectively has unlimited supply) and computes offer prices with the goal of maximizing profit.

There are three potential issues when using this approach: correctness, truthfulness, and performance.

**Correctness.** The technique is correct if it produces a feasible outcome. A mechanism for the unlimited-supply case, $\mathcal{M}_2$, could output any subset of $H$ as its final outcome; this immediately imposes the constraint that the set of feasible outcomes must be closed under inclusion. This condition, which is satisfied by the knapsack problem, is also sufficient as asserted by the following lemma.

**Lemma 5.6.1** *If the set of feasible outcomes are closed under inclusion and $\mathcal{M}_1$ produces a feasible outcome then $\mathcal{M}_1 \circ \mathcal{M}_2$ produces a feasible outcome.*

**Truthfulness.** We would also like the construction to yield a truthful mechanism. Unfortunately, the condition that $\mathcal{M}_1$ and $\mathcal{M}_2$ be truthful is not enough to guarantee the truthfulness of the composite mechanism. In this discussion, we refer to the agents in $H$ (Definition 5.4) as the *survivors* and the prices offered by Step 1 as the *survival prices*. Note that if $\mathcal{M}_1$ is truthful, the survival price of an agent does not depend on her bid. However, even for a truthful mechanism $\mathcal{M}_1$, the winner set $H$ could be a function of some survivor's bid value. In this case, that agent can manipulate her bid to change the set $H$ which may affect the price she is offered by $\mathcal{M}_2$. Thus, we must require that $\mathcal{M}_1$ satisfy a stronger property than truthfulness.

**Definition 5.5 (Composability)** *A mechanism is* composable *if it is truthful and the survivor set produced does not change as a winning agent's bid varies above her survival price.*

**Lemma 5.6.2** *Algorithm* APPROX-KNAPSACK *is composable.*

**Proof:** First we show truthfulness, then we show composability. For truthfulness, note that Algorithm APPROX-KNAPSACK specifies a *monotone allocation rule*, which means that with all other agents' bids fixed, if a winning agent raises her bid, she continues to be in the winning set. For such a monotone allocation rule, the truthful payment rule is to have each agent pay the minimum value that they could bid to be selected. For winning agent $i$, this value is precisely $d^* c_i$ as set by the algorithm.

For composability, we need to show that when the bids of all the agents except one are fixed arbitrarily, the set of selected items as a function of this one agent's bid is unchanged for all the winning bid values of this agent. Whenever the Algorithm APPROX-KNAPSACK selects agent $i$, the other agents selected are exactly those that would have been selected had we run the algorithm without agent $i$ on a knapsack of size $C' = C - c_i$ (after ignoring agents with size greater than $C/2$). Since agent $i$ cannot affect the outcome of this process, the algorithm is composable. □

The rationale for the term "composable" comes from the following lemma.[2]

[2]Note that composability plays a role similar to *cancellability* in Fiat et al. [FGHK02]. In a cancellable auction, the auction's profit is not a function of the value of any winning bid. This allows the auction to be canceled as a function of its profit.

**Lemma 5.6.3** *If mechanism $\mathcal{M}_1$ is composable and mechanism $\mathcal{M}_2$ is truthful then the composite mechanism, $\mathcal{M}_1 \circ \mathcal{M}_2$, is truthful.*

**Performance.** The final issue in using this composition technique to reduce a limited-supply optimization problem to an unlimited-supply optimization problem is to ensure that it has good performance. Given some benchmark for gauging performance on the full input, the feasible outcome produced by $\mathcal{M}_1$ must not limit the possible solutions to ones that are substantially worse, in terms of the chosen benchmark, than the optimal solution on the full input. If this is the case, then with an approximately-optimal unlimited-supply mechanism, $\mathcal{M}_2$, the composite mechanism approximates the chosen benchmark on the full input.

Recall the definition of *performance preservation* (Definition 5.3), and Lemma 5.5.7 which asserts that Algorithm APPROX-KNAPSACK approximates preserves the revenue of the optimal fractional monotone pricing. This makes Algorithm APPROX-KNAPSACK a good candidate for $\mathcal{M}_1$. The missing ingredient thus far is an approximately-optimal unlimited-supply knapsack auction that can be used as $\mathcal{M}_2$. We present such an auction in the next section. We combine Lemma 5.5.7 with Theorem 5.6.10, which shows that the unlimited-supply knapsack auction that we present in the next section approximates the optimal monotone pricing, to get the following theorem.

**Theorem 5.6.4** *Let $OPT$ be the payoff the optimal fractional limited-supply monotone pricing function. Then the payoff of the composite mechanism obtained from Auction* APPROX-KNAPSACK *and Auction* UNLIMITED-SUPPLY-KNAPSACK *is at least*

$$\alpha \,\mathrm{OPT} - \gamma h \lg\lg\lg n$$

*for some constants $\alpha$ and $\gamma$.*

Our auction can be modified slightly to get a payoff of at least $\alpha OPT_{n'} - \gamma h \lg\lg\lg n'$, where $OPT_{n'}$ is the optimal monotone payoff from at most $n'$ winners for any choice of $n'$.

### 5.6.2 Unlimited-Supply Knapsack Auction

In this section, we consider the knapsack auction problem when $C = \infty$. We first attempt to use the general attribute auction of Blum et al. [BH05] to solve this problem. Since the optimal monotone pricing rule might offer a different price to every agent, the number of piecewise-constant pieces needed to emulate this rule could be as high as the number of agents $n$. Thus, a direct application of the attribute auction result (Theorem 5.3.1) to the knapsack auction problem would only guarantee a minimum profit of $\mathrm{OPT}/16 - nh/2 \leq 0$, where $OPT \leq nh$ is the payoff of the optimal monotone pricing rule. Still, the unlimited-supply knapsack auction problem remains closely related to the attribute auction problem, and we will be making use of Theorem 5.3.1 in this section.

Let $n'$ be the number of winners for the optimal monotone pricing function. Our results come from observing Lemma 5.6.5 below, which implies that there is a monotone pricing function with close to optimal payoff that

(a) divides the size range into $\lg n'$ intervals and for each interval, offers the same price to all agents whose size lies in the interval, and

(b) most (all but $O(\lg\lg\lg n)$) of the intervals have many (at least $O(\lg\lg n)$) winners.

Simply using part (a) of this fact and applying the result of Blum and Hartline [BH05], we can obtain an auction that is $\mathrm{OPT}/32 - h\lg n'/2$. The main result of this section will be to use part (b) of this fact to improve the additive loss term to $O(h\lg\lg\lg n)$.

We obtain this result by analyzing two possible cases. In the first case, most of the payoff of the optimal monotone rule with exponential intervals (see Definition 5.6 below) comes from the large intervals with at least $\Omega(\lg\lg n)$ winners. For these large intervals, we can apply random sampling techniques and the Chernoff bound to show that a generalization of the random sampling auction [GHW01] that uses the optimal pricing rule with exponential intervals will obtain a constant fraction of the optimal monotone payoff.

On the other hand, if most of the payoff of the optimal monotone rule with these properties comes from the $\Theta(\lg\lg\lg n)$ small intervals, then the result of Blum and Hartline can be applied to get an auction that obtains a constant fraction of $\mathrm{OPT}$ less an additive term that is linear in the number of relevant intervals. This gives an additive loss term of $O(h\lg\lg\lg n)$.

A convex combination of these two techniques gives the auction below. We start with a definition and a lemma.

**Definition 5.6** *A* monotone pricing rule with exponential intervals *is a monotone pricing rule in which the winners can be partitioned into equal-priced intervals over the attributes such that the $i^{th}$ interval (in decreasing order of attribute value) contains at least $2^{i-1}$ winners, unless $i$ is the last interval.*

**Lemma 5.6.5** *Given any monotone pricing rule, $\pi(\cdot)$, that obtains total payoff $P$ on instance $(v_1, \ldots, v_n; c_1, \ldots, c_n; C = \infty)$, there is a monotone pricing rule with exponential intervals, $\pi'(\cdot)$ with payoff at least $P/2$.*

**Proof:** Order the winners of $\pi(\cdot)$ on the instance by decreasing size (breaking ties arbitrarily). Divide the attribute range into intervals such that the $i^{th}$ interval has at least $2^{i-1}$ winners but strictly fewer than $2^{i-1}$ winners with size strictly bigger than the smallest winner in $i$. This can be done by considering the attributes in decreasing order and adding them to the current interval until the first time the number of winners in the interval becomes at least $2^{i-1}$. At this point, we move on to the next interval. Let $c(i)$ be the size of this smallest item in interval $i$. Consider $\pi'(\cdot)$ defined such that all items in interval $i$ are offered price $\pi(c(i))$.

Now we show that the payoff of $\pi(\cdot)$ is no more than twice that of $\pi'(\cdot)$. The loss for interval $i$ is the difference in payoff between $\pi'(\cdot)$ and $\pi(\cdot)$ over the attribute interval $[c(i), c(i-1))$. There is no loss from items with size exactly $c(i)$ and the loss from other items in interval $i$ is bounded by $\pi(c(i-1)) - \pi(c(i))$. Since interval $i$ contains fewer than $2^{i-1}$ items with size strictly more than $c(i)$, the total loss is no more than $(2^{i-1} - 1) \times (\pi(c(i-1)) - \pi(c(i)))$. We charge this loss to the winners in all the previous intervals. There are at least $\sum_{j=1}^{i-1} 2^{j-1} = 2^{i-1} - 1$ such winners; so each winner is charged at most $\pi(c(i-1) - \pi(c(i))$. Now consider the total amount charged to a winner in interval $i$ by subsequent intervals. The charges made to any given winner in interval $i$ telescope and sum to at most $\pi(c(i))$; thus the total loss can be accounted for by the total payoff of $\pi'(\cdot)$. Therefore the payoff of $\pi'(\cdot)$ is at least half that of $\pi(\cdot)$. □

Now, we are ready to define the random-sampling part of the unlimited-supply auction.

**Auction** RANDOM-SAMPLING-KNAPSACK or RSK

1. Partition the agents into two sets $A$ and $B$ uniformly at random.
2. Compute the optimal 'monotone pricing rule with exponential intervals (restricting prices to powers of two) for each partition. Let the pricing rules for $A$ and $B$ be $\pi_A$ and $\pi_B$ respectively.
3. Use $\pi_A$, the pricing rule for $A$, to offer prices to set $B$ and vice versa.

**Lemma 5.6.6** *Auction* RANDOM-SAMPLING-KNAPSACK *or RSK is truthful.*

**Proof:** Recall the definition of a bid-independent auction (Definition 5.1). Theorem 5.1.1 implies that if the price offered to each agent is independent of her bid, then the auction is truthful. Now note that the price offered by RSK to an agent $i$ (say in set A) depends only the bids of the agents in set B, and does not depend on her bid at all. □

Let $\pi_A$ on $A$ have $n_A$ winners. Let $\bar{n}_A$ be the largest power of 2 that is no larger than $n_A$. Then, the winners are divided up into at most $\lg \bar{n}_A + 1$ equal-priced markets. A market is said to be *large* for $A$ if it has at least $256 \lg \lg \bar{n}_A$ winners when $\pi_A$ is applied to A. Note that all markets other than the first $\lg \lg \lg n_A + 8$ markets and the last market (by decreasing attribute value) are large. Markets that are not large are called *small*. We wish to analyze the performance of RSK on the large markets. Define $\pi'_A$ to be a the pricing rule that is the same as $\pi_A$, except that it offers a price of $\infty$ to the small markets. Let $P(\pi, A)$ denote the total profit of pricing function $\pi$ applied to set $A$. Let $\mathcal{L}$ be an ordering of the agents in the decreasing order of attribute value (breaking ties arbitrarily). Let $\mathcal{L}_p$ denote the ordering $\mathcal{L}$ restricted to agents having bids $p$ or higher. We first prove a few useful lemmas.

**Definition 5.7 (Bad Event, Bad Set, Potential Bad Set)** *A* Bad Event *is said to have occurred in RSK if there exists an $\eta = 2^k$ for integer $k \geq 4$, a price $p = 2^r$ for integer $r$ s.t. $\lg h/\eta^2 < r \leq \lg h$, and a subset $X$ of agents, satisfying the following properties:*

*(i) All the agents in $X$ have bids $p$ or higher, and appear consecutively in $\mathcal{L}_p$.*

*(ii) $|X| \geq \frac{3}{2} \max\left\{\frac{m_p}{6 \lg \eta}, 256 \lg \lg \eta\right\}$, where $m_p$ is the total number of agents with bid $p$ or higher.*

*(iii) One of the two sets created by RSK has more than $2|X|/3$ of the agents in $X$.*

*A set $X$ that satisfies the first two properties is called a* Potential Bad Set, *while any set $X$ that satisfies all the above properties is called a* Bad Set.

**Lemma 5.6.7** *The probability of a Bad Event occurring in Auction* RANDOM-SAMPLING-KNAPSACK *is no more than* $0.01$.

**Proof:** We will prove that the probability of the existence of a Bad Set $X$ for which set $A$ gets more than $2|X|/3$ of the agents is no more than $0.005$. By symmetry, the probability of the existence of a Bad Set $X$ w.r.t. $B$ is also no more than $0.005$. Then, we can take the union bound to get the lemma. We first prove the following statement:
**Claim:** Consider a set $X$ of $3x$ agents. The probability that set $A$ gets more than $2x$ agents is no more than $e^{-x/12}$.
**Proof:** The expected number of agents from $X$ that fall into set $A$ is $1.5x$. By Chernoff bounds [MR95], the probability that fewer than $x$ agents fall into it is no more than $e^{\frac{-(1.5x)(1/3)^2}{2}} = e^{-x/12}$.

We now use this claim along with a series of union bounds to prove the lemma. Fix a number $\eta = 2^k$ for some integer $k \geq 4$ and a price $p = 2^r$ for some $r$ s.t. $\lg h/\eta \leq r \leq \lg h$. Let $m_p$ be the total number of agents with price $p$ or higher. Arrange these agents by decreasing order of attribute value. Let $L_\eta = \max\{\frac{m_p}{6 \lg \eta}, 256 \lg \lg \eta\}$. Consider a subset $X$ of $3x$ consecutive agents where $2x \geq L_\eta$. By the above claim, the probability that this subset splits such that set $A$ has more than $2x$ of these agents is no more than $e^{-x/12}$. Taking the union bound, the probability of such a subset existing for these fixed values of $n$ and $p$ is no more than

$$\begin{aligned}
& m_p \sum_{|X|=3L_\eta/2}^{m_p} e^{-|X|/36} \\
\leq\ & m_p e^{-L_\eta/24}/(1 - e^{-1/36}) \\
\leq\ & 36.6 * (6L_\eta \lg \eta) e^{-L_\eta/24} \\
\leq\ & 220 L_\eta 2^{L_\eta/256} 2^{-1.44 L_\eta/24} \\
=\ & (220 L_\eta 2^{-L_\eta/30}) 2^{-1.44L_\eta/24 + L_\eta/256 + L_\eta/30} \\
\leq\ & e^{-L_\eta/44}
\end{aligned}$$

To get the last inequality, we used the fact that $L_\eta \geq 256 \lg\lg\eta \geq 512$ when $n \geq 2^4$. Taking the union bound over all possible values of $p$ (there are at most $2\lg\eta$ of them), we get that the probability of such a subset existing for a given value of $\eta$ is no more than

$$\begin{aligned} & 2(\lg\eta)2^{-L_\eta/44} \\ \leq\ & 2(2^{\frac{256\lg\lg\eta}{44}}\lg\eta) \\ \leq\ & 2(\lg n)^{-\frac{256}{44}+1} \\ \leq\ & 2(\lg\eta)^{-4.8} \end{aligned}$$

Taking the union bound over all $\eta = 2^k$ for $k = 4, 5, \cdots$, we get that the probability is no more than

$$\begin{aligned} & 2\sum_{k=4}^{\infty} k^{-4.8} \\ =\ & 2\left(\frac{1}{4^{4.8}} + \frac{1}{5^{4.8}} + \frac{1}{6^{4.8}} + \cdots\right) \\ \leq\ & 0.005 \end{aligned}$$

The inequality is obtained by using an integral to approximate the summation. □

We now prove the following lemma about the revenue of Auction RANDOM-SAMPLING-KNAPSACK. A similar lemma holds when the roles of $A$ and $B$ are interchanged.

**Lemma 5.6.8** *For Auction* RANDOM-SAMPLING-KNAPSACK,

$$\mathbf{E}\left[P(\pi'_A, B)\right] \geq \max\left\{0, \beta\left(\mathbf{E}\left[P(\pi'_A, A)\right] - \frac{1}{2}\mathbf{E}\left[P(\pi_A, A)\right] - \frac{h}{2}\right)\right\}$$

*for* $\beta = 0.99/2$.

**Proof:** If $\bar{n}_A < 16$, $\mathbf{E}\left[P(\pi'_A, A)\right] = 0$, making the claim trivially true. Thus, we will assume that $\bar{n}_A \geq 16$. We will also assume that $\mathbf{E}\left[P(\pi_A, A)\right] > h$; otherwise the claim is trivially true.

Recall that a market is *large* for $A$ if it has at least $256\lg\lg\bar{n}_A$ winners when $\pi_A$ is applied to A. If $\pi_A$ applied to a large market has a profit greater than $\frac{\mathbf{E}[P(\pi_A,A)]}{2\lg\bar{n}_A}$, then that

market is called *significant* for A. Since the number of large markets is at most $\lg \bar{n}_A$, the total profit on applying $\pi'_A$ to the significant markets of $A$ is at least $\max\{0, \mathbf{E}[P(\pi'_A, A)] - \mathbf{E}[P(\pi_A, A)]/2\}$. Thus, we can prove the lemma by showing that with constant probability, $\mathbf{E}[P(\pi'_A, B)]$ is at least a constant fraction of the profit from applying $\pi'_A$ to the significant markets of $A$.

Let $n_i(\pi_A, A)$ denote the number of winners in the $i^{th}$ market when $\pi_A$ is applied to $A$, while $n_i(\pi_A, B)$ denote the number of winners in the $i^{th}$ market when $\pi_A$ is applied to $B$. We will show that assuming no *Bad Event* (see Definition 5.7) has occurred, there is no significant market $i$ of $A$, s.t. $n_i(\pi_A, A) > 2n_i(\pi_A, B)$. Since no Bad Event occurs with probability at least $0.99$, it would immediately imply that

$$\mathbf{E}[P(\pi'_A, B)] \geq \max\{0, \frac{0.99}{2}(\mathbf{E}[P(\pi'_A, A)] - \frac{1}{2}\mathbf{E}[P(\pi_A, A)])\}.$$

Assume that no Bad Event has occurred. For a contradiction, suppose that there is a significant large market $i$ of $A$ that has $n_i(\pi_A, A) > 2n_i(\pi_A, B)$. Let $[a, b]$ be the attribute range corresponding to this market. Also let price offered to the $i^{th}$ market by $\pi_A$ be $p = 2^k$ for some integer $k$ and $m_p$ be the total number of agents with bid $p$ or higher. We claim that $p > \frac{h}{n_A^2}$. Suppose to the contrary, the price $p \leq \frac{h}{n_A^2}$. Then the $i^{th}$ market has a payoff of at most

$$\begin{aligned} \frac{h}{n_A} &< \frac{h}{2\lg \bar{n}_A} \\ &< \frac{\mathbf{E}[P(\pi_A, A)]}{2\lg \bar{n}_A} \end{aligned}$$

This would imply that market $i$ is not a significant market, a contradiction to the supposition above.

Recall that $\mathcal{L}_p$ is an ordering of all the agents with bids $p$ or higher in decreasing order of attribute value. Consider a set $X$ of $\frac{3}{2}n_i(\pi_A, A)$ agents with bids $p$ or higher who appear consecutively in $\mathcal{L}_p$, such that $X$ includes all agents in the attribute range $[a, b]$. Then, by assumption, more than $\frac{2}{3}|X|$ agents from this set are in set $A$. We show that $X$ is a *Potential Bad Set* with $\eta = \bar{n}_A$. We already know that $|X| \geq \frac{3}{2}(256 \lg\lg \bar{n}_A)$. Thus, all we need to show is that $|X| \geq \frac{m_p}{4\lg \bar{n}_A}$, or alternatively, that $n_i(\pi_A, A) \geq \frac{m_p}{6\lg \bar{n}_A}$. To

see this, note that since market $i$ is significant for A, $n_i(\pi_A, A)p > \frac{P(\pi_A,A)}{2\lg \bar{n}_A}$. In other words, $(2n_i(\pi_A, A)\lg \bar{n}_A)p > P(\pi_A, A)$. If more than $2n_i(\pi_A, A)\lg \bar{n}_A$ agents in set $A$ had bids of $p$ or higher, then offering a price of $p$ to everybody would yield a profit of more than $P(\pi_A, A)$, contradicting the optimality of $\pi_A$ for set $A$. Thus, the number of agents in set $A$ with bid $p$ or higher is no more than $2n_i(\pi_A, A)\lg \bar{n}_A$. Consider the set of all agents with bids $p$ or higher. This is a Potential Bad Set. Since the Bad Event has not occurred, the third condition for a Bad Event is not satisfied. Thus, if set $A$ has no more than $2n_i(\pi_A, A)\lg \bar{n}_A$ agents with bid $p$ or higher, then the total number of agents with bid $p$ or higher $m_p \leq 3(2n_i(\pi_A, A)\lg \bar{n}_A)$, or $n_i(\pi_A, A) \geq \frac{m_p}{6\lg \bar{n}_A}$, as required.

Thus, the set $X$ is a Potential Bad Set. Since the Bad Event has not occurred, set $X$ does not satisfy the third condition of being a Bad Set, implying that the number of agents from set $X$ in set $A$ is no more than $\frac{2}{3}|X|$, thus contradicting the supposition that $n_i(\pi_A, A) > 2n_i(\pi_A, B)$. □

Now consider the following combination of the two auctions.

**Auction** UNLIMITED-SUPPLY-KNAPSACK

1. Perform Step 1 of Auction RANDOM-SAMPLING-KNAPSACK.

2. With probability $p$, run the general attribute auction on the sets $A$ and $B$ separately. With the remaining probability, run the remaining steps of Auction RANDOM-SAMPLING-KNAPSACK.

It is easy to see that since RSK and the general attribute auction are truthful (and thus bid-independent), the above auction is also bid-independent. Then, using Theorem 5.1.1, we get the following lemma.

**Lemma 5.6.9** *Auction* UNLIMITED-SUPPLY-KNAPSACK *is truthful.*

**Theorem 5.6.10** *The revenue generated by Auction* UNLIMITED-SUPPLY-KNAPSACK *is* $\alpha\,\mathrm{OPT} - \gamma h(\lg\lg\lg n_A + \lg\lg\lg n_B + 19)$*, where* OPT *is the payoff of the optimal monotone pricing and* $\alpha$ *and* $\gamma$ *are constants.*

**Proof:** Recall that $OPT$ is the payoff of the optimal monotone pricing scheme. Using Lemma 5.6.5 and losing another factor of 2 due to restriction to prices that are powers of 2, $\mathrm{OPT} \leq 4(\mathbf{E}[P(\pi_A, A)] + \mathbf{E}[P(\pi_B, B))]$.

Recall that any market with fewer than $256 \lg\lg\lg \bar{n}_A$ winners is small for $A$. There are at most $\lg\lg\lg \bar{n}_A + 9$ small markets of A with respect to $\pi_A$. Similarly, there are at most $\lg\lg\lg \bar{n}_A + 9$ small markets of B with respect to $\pi_B$. Note that markets that are not small are *large* for their respective sets. Let $P(\pi_A, A_S)$ be the payoff of applying $\pi_A$ to the small markets of $A$. Define $P(\pi_B, B_S)$ similarly. Then, $\mathbf{E}[P(\pi_A, A)] + \mathbf{E}[P(\pi_B, B)] = \mathbf{E}[P(\pi_A, A_S)] + \mathbf{E}[P(\pi_B, B_S)] + \mathbf{E}[P(\pi'_A, A)] + \mathbf{E}[P(\pi'_B, B)]$.

With probability $p$, we use the general attribute auction, in which case, by Theorem 5.3.1, we get an expected revenue of at least

$$(\mathbf{E}[P(\pi_A, A_S)] + \mathbf{E}[P(\pi_B, B_S)])/16 - \frac{h}{2}(\lg\lg\lg n_A + \lg\lg\lg n_B + 18).$$

On the other hand, when we use Auction RANDOM-SAMPLING-KNAPSACK (which we do with probability $(1-p)$), we can apply Lemma 5.6.8 and the same lemma with $A$ and $B$ interchanged, to show that we get an expected revenue of at least

$$\beta\left(\mathbf{E}[P(\pi'_A, A)] - \frac{1}{2}\mathbf{E}[P(\pi_A, A)] + \mathbf{E}[P(\pi'_B, B)] - \frac{1}{2}\mathbf{E}[P(\pi_B, B)] - h\right)$$

for $\beta = 0.99/2$. Thus, the overall expected revenue is at least

$$\begin{aligned}
&\frac{p}{16}\left(\mathbf{E}[P(\pi_A, A_S)] + \mathbf{E}[P(\pi_B, B_S)]\right) \\
&\quad - \frac{ph}{2}\left(\lg\lg\lg n_A + \lg\lg\lg n_B + 18\right) \\
&\quad + (1-p)\beta\left(\mathbf{E}[P(\pi'_A, A)] + \mathbf{E}[P(\pi'_B, B)] - \frac{\mathbf{E}[P(\pi_A, A)]}{2} - \frac{\mathbf{E}[P(\pi_B, B)]}{2} - h\right)
\end{aligned}$$

Setting $p = \frac{24\beta}{24\beta+1}$, we get an expected revenue of at least

$$\begin{aligned}
&\frac{3\beta}{2(24\beta+1)}\left(\mathbf{E}\left[P(\pi_A, A_S)\right] + \mathbf{E}\left[P(\pi_B, B_S)\right]\right) \\
&\qquad - \frac{12\beta h}{24\beta+1}\left(\lg\lg\lg n_A + \lg\lg\lg n_B + 18\right) \\
&\qquad + \frac{\beta}{24\beta+1}\left(\mathbf{E}\left[P(\pi'_A, A)\right] + \mathbf{E}\left[P(\pi'_B, B)\right] - \frac{\mathbf{E}\left[P(\pi_A, A)\right]}{2} - \frac{\mathbf{E}\left[P(\pi_B, B)\right]}{2} - h\right) \\
\geq\ &\frac{\beta}{24\beta+1}\left(\mathbf{E}\left[P(\pi_A, A)\right] + \mathbf{E}\left[P(\pi_B, B)\right]\right) \\
&\qquad - \frac{12\beta h}{24\beta+1}\left(\lg\lg\lg n_A + \lg\lg\lg n_B + 18 + \frac{1}{12}\right)
\end{aligned}$$

To get the inequality, we have used the fact that $\mathbf{E}\left[P(\pi_A, A)\right] + \mathbf{E}\left[P(\pi_B, B)\right] \leq \mathbf{E}\left[P(\pi_A, A_S)\right] + \mathbf{E}\left[P(\pi_B, B_S)\right] + \mathbf{E}\left[P(\pi'_A, A)\right] + \mathbf{E}\left[P(\pi'_B, B)\right]$. Putting in the value $\beta = 0.99/2$, and using the fact that $\mathrm{OPT} \leq 4(\mathbf{E}\left[P(\pi_A, A)\right] + \mathbf{E}\left[P(\pi_B, B))\right]$, we get the theorem with $\alpha = 0.009$ and $\gamma = 0.47$. □

Noting that $n_A$ and $n_B$ are no more than $n$, we get Theorem 5.6.4. Auction RANDOM-SAMPLING-KNAPSACK can be modified in order to achieve a revenue of at least $\alpha\,\mathrm{OPT}(n') - \gamma h(2\lg\lg\lg n' + 19)$, where $\mathrm{OPT}(n')$ is the payoff of the optimal monotone pricing with no more than $n'$ winners and $\alpha$ and $\gamma$ are the same constants as above. For this, we change Step 2 of Auction RANDOM-SAMPLING-KNAPSACK and for each of the two sets, compute the optimal monotone pricing rule with exponential intervals (with prices restricted to powers of 2) that has at most $n'$ winners. The rest of the auction proceeds as before.

# Chapter 6

# The Search Engine Problem

Targeted advertising is an indispensable part of the business model of modern web search engines and is responsible for a significant share of their revenue. For every keyword, several advertisements are displayed beside the search results, and the visibility of an advertisement depends on its location (slot) on the web page. Typically, a number of merchants are interested in advertising alongside each keyword and they naturally prefer slots with higher visibility. Even if the number of merchants interested in a keyword is no more than the number of available slots, the search engine has to match merchants to display locations. However, if the number of merchants is more than the number of advertisement slots available, the search engine also has to pick a subset of advertisements relevant to the keyword. In addition, the search engine has to decide on a price to be charged to each merchant. Due to the dynamic nature of the advertising market, most search engines are using auctions to solve the problem of selling advertisement space alongside keyword search results. In a keyword auction, merchants are invited to submit bids on the keyword, i.e. the maximum amount they are willing to pay for an Internet user clicking on their advertisement. Typically, the search engines charge a merchant only when a user actually clicks on the advertisement. Based on the bids submitted for the keyword, the search engine (which we will sometimes refer to as the auctioneer) picks a subset of advertisements along with the display order. The price charged also depends on the set of submitted bids.

In the auctions currently being used, the search engine first picks the subset of advertisements to be displayed and matches them to slots based on the submitted bids; the matching

criteria is referred to as the *ranking function* and is an integral component of the existing keyword auctions. Then, the auctioneer decides on a price for each merchant based on the bids and the allocation. There are two popular ranking methods:

1. The *Overture* (or Yahoo!) method: Merchants are ranked in the decreasing order of the submitted bids. We will call this *direct ranking*.

2. The *Google* method: Merchants are ranked in the decreasing order of the *ranking scores*, where the ranking score of a merchant is defined as the product of the merchant's bid and estimated click-through rate. We will refer to this as *revenue ranking*.

These ranking functions are an inherent part of the advertisement philosophies of Overture and Google respectively. Accordingly, we will assume that these ranking functions are fixed. Hence the only degree of freedom in running the auction is the price charged per click-through to each merchant. Both Overture and Google currently charge a merchant the minimum amount she would need to bid to retain its current rank in the auction[1]. This price can never be larger than the submitted bid, since clearly, the submitted bid was enough to guarantee the merchant her current rank. The utility of a merchant is her expected gain, i.e., the difference between the benefit she receives and the price she pays for it; it is defined more formally in Section 6.1. We will refer to this auction as the *next-price* auction. Despite superficial similarity to the truthful second-price auction [Vic61] described in Section 4.3, the next-price auction is not truthful — in Section 6.2.1, we present examples where a merchant has an incentive to bid less than her true valuation under the above auctions. We observe that in the current auctions run by Google and Overture, there is an asymmetric incentive for merchants — there may be an incentive for a merchant to bid less than her true valuation for each click on her advertisement, but there is never an incentive for her to bid more than her true valuation.

Since truth-telling is not a dominant strategy in the current auctions, there is no clear prescription for merchants to determine their optimum bid. The optimum bid depends in a complicated and dynamic manner on externalities such as the bids of the other merchants, and it is often necessary for merchants to hire expensive consultants or intermediaries to

[1] Plus a fixed small increment, but we will ignore this minor detail.

formulate their bidding strategies. As mentioned in the introduction, the lack of clear bidding strategies is slowing the growth of online advertising. A truthful mechanism would simplify the bidding process significantly, since it would require a merchant to only determine her valuation for the keyword, a quantity that is intrinsic to the merchant. A truthful mechanism would also remove the incentive for a merchant to under-bid. Furthermore, in the case of revenue ranking and with an additional separability assumption (defined in Section 6.1), a truthful mechanism is *efficient* in the sense that it maximizes the total (weighted) utility obtained by the auctioneer and the merchants together. This motivates us to study the problem of designing truthful keyword auctions.

One might be tempted to suggest that the famous VCG mechanism [Vic61, Cla71, Gro73] (see Chapter 4 for a description) or a weighted and biased variant of it would yield a solution to this problem. However, we give an example (Section 6.2.2) where there does not exist any set of weights and biases for which the VCG mechanism always outputs the same merchant ordering as the given ranking function. Hence, the VCG mechanism is not generally applicable to our problem. We further discuss the applicability of VCG in Section 6.2.

**Our Contribution.** We design a simple truthful auction for a general class of ranking functions that includes direct ranking and revenue ranking. A ranking function in this class assigns an a priori weight to each merchant that is independent of her bid and then ranks the merchants in the decreasing order of their weighted bids — this is defined formally in Section 6.1. In particular, setting all the weights to 1 results in the direct ranking used by Overture, and setting the weights equal to the estimated click-through rates results in the revenue-ranking scheme used by Google.

We call our auction the *laddered auction*, since the price charged to a merchant builds on the price charged to merchants ranked below it. We show that this auction is truthful. Further, we show that the laddered auction is the unique truthful auction, and hence is trivially revenue-maximal for the auctioneer among all truthful auctions. The auction is presented in Section 6.3 and the analysis is in Section 6.4.

**Revenue Equivalence.** We then ask the next natural question: how will the auctioneer's revenue change as a result of implementing our truthful auction rather than the next-price auction currently in use? Since the next-price auction is not truthful, its revenue should be computed assuming that the bids of the merchants are in a Nash equilibrium, i.e., the bids are such that no merchant can increase her profit by a unilateral change in her bid. For arbitrary click-through rates, we have not been able to answer this question, primarily because we can not obtain a simple characterization of the Nash equilibria imposed by the next-price auction in this case. However, when the click-through rates are separable (i.e. the click-through rates can be separated into a merchant-specific factor and a position-specific factor; see Section 6.1 for a formal definition), we prove the following revenue-equivalence theorem:

> There exists a *pure-strategy* Nash equilibrium for the next-price auction which yields exactly the same revenue for the auctioneer as our laddered auction.

We give an explicit characterization of this Nash equilibrium. These results are presented in Section 6.5. Interestingly, we show that there may exist other pure-strategy Nash equilibria under which the next-price auction achieves a smaller revenue than the truthful auction, and yet others under which the next-price auction achieves a higher revenue; these examples are presented in Section 6.5.1. In fact, starting from the truthful bids, there may be sequences of self-interested moves (i.e. bid changes) that can lead to a Nash equilibrium for the next-price auction of higher or lower revenue than the truthful auction. This suggests that while the revenue of the current auctions could be better or worse than the truthful auction depending on which equilibrium the bids settle into, the revenue of our truthful auction is more predictable.

**Discussion and Related Work.** We assume throughout that the number of slots, $K$, that the auctioneer sells for a given keyword does not depend on the submitted bids (although it may depend on the number of merchants taking part in the auction). Our auction is not truthful if the auctioneer computes the optimum number of slots (in terms of the revenue generated by our laddered pricing scheme) to be displayed. Extending our auctions to this case appears to be a non-trivial and interesting research direction.

Since the submitted bids are typically used for more than one impression, in addition to a merchant's valuation, her budget may also be a parameter of relevance [MSVV05, BCI$^+$05]. However, in many cases, merchants want users to go to their web sites to purchase merchandise which results in immediate profit; in this case, a merchant's true valuation for a keyword is simply her expected immediate profit per click. If the merchant bids in accordance with the truth-telling strategy that is dominant for our laddered auction, she would bid an amount equal to her valuation of the keyword. Since the price charged by the auctioneer is never larger than the bid, each click results in an immediate net profit. Hence, ignoring budgets would be the right thing to do under this scenario.

Some recent work [MSVV05, BCI$^+$05] studies the web advertisement problem with budget constraints. Mehta et al. [MSVV05] ignore the game-theoretic issues and instead focus on the algorithmic problem of matching merchants to web pages when their valuations and budgets are known to the auctioneer. Borgs et al. [BCI$^+$05] study the problem of selling multiple identical units when the agents are interested in getting multiple units as long as their payment does not exceed their budget. While a model with multiple identical units might be applicable to the case of web pages with a single advertisement slot, it is not suitable for web pages with multiple advertisement slots, as it does not take into account the inherent differences in visibility between various positions (slots) on the same page.

## 6.1 Model and Notation

There are $N$ merchants bidding for $K < N$ slots on a specific keyword (if $K \geq N$, reduce the number of slots $K$ to $N$ and add a dummy merchant with all relevant parameters set to 0). The slots are numbered in the order of decreasing visibility, i.e. the visibility of the slot numbered $i$ is no less than that of the slot numbered $i + 1$ for all $i$. Let $\text{CTR}_{i,j}$ denote the click-through rate of the $i^{th}$ merchant if placed at slot $j \leq K$. We assume that $\text{CTR}_{i,j}$ is arbitrary, but known to the auctioneer. Also, we assume that $\text{CTR}_{i,j}$ is non-increasing in $j$. Set $\text{CTR}_{i,j} = 0$ for $j > K$. Let $v_i$ denote the true valuation of a click-through to merchant $i$. (This is again an abuse of notation: we are using $v_i$ to refer to the valuation of a single click-through for merchant $i$, rather than the valuation function of merchant $i$.) We assume that $v_i$ is known to merchant $i$, but not to the auctioneer.

As outlined in the introduction, we will assume that the ranking function is externally specified. We will consider the class of ranking functions where merchant $i$ is assigned an a priori weight $w_i$ that is independent of her bid. Let $b_i$ denote the bid of the $i^{th}$ merchant for each click-through. The merchants are ranked in the order of decreasing $w_i b_i$. Setting $w_i = 1$ for all $i$ is equivalent to the direct ranking function (the Overture model), while setting $w_i = \text{CTR}_{i,1}$ reduces to the revenue-ranking function (the Google model). Merchant $i$ is charged a price-per-click, $p_i \leq b_i$, which is determined by the auction. We assume the merchants to be risk-neutral. As such, if merchant $i$ is placed at position $j$, it obtains a utility of $\text{CTR}_{i,j} \cdot (v_i - p_i)$ per impression. Recall that an auction is truthful if bidding her true valuation (i.e. $b_i = v_i$) is a dominant strategy for every agent (see Chapter 4 for more background on truthful auctions). Now, we formally define the next-price auctions currently being used.

**Definition 6.1 (Next-price Auction)** *Given the ranking function, $R = (w_1, w_2, \ldots, w_n)$ and the bid vector $\mathbf{b} = (b_1, \ldots, b_n)$, the next-price auction ranks the merchants in the decreasing order of $w_i b_i$ and charges the merchant ranked $i$ an amount-per-click equal to the minimum bid she needs to have submitted in order to retain rank $i$. Let $w_a$ and $w_b$ refer to the weights of the merchants ranked $i$ and $i+1$ respectively. And let $b_b$ refer to the bid submitted by the merchant ranked $i+1$. Then the price charged to the merchant ranked $i$ is $\frac{w_b b_b}{w_a}$.*

We will now describe the separability assumption, which we will use (only) for our results on revenue-equivalence. Informally, this assumption states that the click-through rates can be separated into a merchant-specific factor and a position-specific factor.

**Definition 6.2 (Separable Click-through Rates)** *The click-through rates are said to be separable if there exist $\mu_1, \mu_2, \ldots \mu_n > 0$ and $\theta_1 \geq \theta_2 \geq \ldots \theta_K > 0$ such that the click-through rate $\text{CTR}_{i,j}$ of the $i^{th}$ merchant at the $j^{th}$ slot is given by $\mu_i \theta_j$.*

There is evidence to believe that this is a reasonable assumption that holds (approximately) in many real-world cases.

## 6.2 Need for a New Auction

In this section, we begin by giving an example to show that the next-price auctions being currently used by Google and Overture are not truthful. In order to construct a truthful auction, the first logical step is to see whether the famous VCG mechanism (see Section 4.4) applies to the problem. However, this is not the case, and we give instances of ranking functions for which there does not exist any set of weights and biases for which the ranking output by the VCG mechanism is always the same as the one output by the given ranking function.

### 6.2.1 Next-price Auction is not Truthful

Consider three merchants $A$, $B$ and $C$ bidding for two slots. Let all three of them have a click-through rate of $0.5$ at the top slot and $0.4$ at the bottom slot. Let the true valuations per click of the three merchants be 200, 180, and 100 respectively. Then, if all the merchants bid truthfully, merchant $A$ ends up paying a price of $180$ per click, making an expected profit of $(200 - 180) \times 0.5 = 10$ per impression. In this case, she has an incentive to undercut $B$ by lowering her bid to $110$, and make a net profit of $(200 - 100) \times 0.4 = 40$. We note that there is no incentive to bid higher than one's true valuation under the next-price auction. This is because the price-per-click charged is the minimum bid required to retain one's rank; therefore, in cases where bidding higher improves one's rank, the price-per-click charged is higher than one's true valuation.

### 6.2.2 Weighted VCG may not Always Apply

In this section, we show by means of a counter-example that even for the simple case of direct ranking, there does not exist any set of (bid-independent) weights and biases for which the VCG solution achieves the same allocation as direct ranking. This will show that, in general, VCG does not apply to our problem. Consider two merchants $A$ and $B$ bidding for two slots on a web page. Let both the merchants have a click-through rate (CTR) of $0.4$ at the first slot. For the second slot, merchant $A$ has a CTR of $0.4$ while merchant $B$ has a CTR of $0.2$. Since any of the merchants can bid the highest and get

the top slot in direct ranking, both the merchants must have non-zero weight in order for weighted VCG to achieve the same allocation as direct ranking. Let $\omega_A > 0$ and $\omega_B > 0$ be the weights assigned by the VCG mechanism to merchants $A$ and $B$ respectively. Denote the bias assigned to ranking merchant $x$ followed by $y$ by $H(x, y)$ for $x, y \in \{A, B\}$, $x \neq y$. Then, the VCG mechanism will rank $B$ before $A$ if $\omega_A(0.4b_A) + \omega_B(0.4b_B) + H(B, A) > \omega_A(0.4b_A) + \omega_B(0.2b_B) + H(A, B)$, which is true whenever $b_B > (H(A, B) - H(B, A))/(0.2\omega_B)$, irrespective of merchant $A$'s bid. On the other hand, the direct ranking scheme will rank $A$ before $B$ whenever $A$'s bid is higher than $B$'s bid. Thus, the VCG mechanism does not apply to this instance. In fact, we show the following general theorem.

**Theorem 6.2.1** *Let the number of merchants with non-zero click-through rates be $n > K$. If the click-through rates are not separable, then there exists a ranking function $R = (w_1, w_2, \ldots, w_n)$ for which there does not exist any set of weights for which unbiased, weighted VCG always yields the same ranking as the ranking function $R$.*

**Proof:** Let $\text{CTR}_{i,j}$ be the click-through rate of merchant with index $i$ at the $j^{th}$ position. First note that if $\text{CTR}_{i,j}/\text{CTR}_{i,j+1} = \text{CTR}_{i',j}/\text{CTR}_{i',j+1}$ for all values of $i, i' \leq n$ and $j \leq K - 1$, then the click-through rates are separable: just set $\mu_i = \text{CTR}_{i,K}$ and $\theta_j = \text{CTR}_{1,j}/\text{CTR}_{1,K}$.

We will show that if for every ranking function $R = (w_1, w_2, \ldots, w_n)$, there exists a set of VCG weights which always yield the same ranking as $R$, then $\text{CTR}_{i,j}/\text{CTR}_{i,j+1} = \text{CTR}_{i',j}/\text{CTR}_{i',j+1}$ for all values of $i, i' \leq n$ and $j \leq K - 1$. We will prove this by downward induction on $j$.

First consider the base case of $j = K - 1$. Consider any pair of merchants. Re-index the merchants such that the two merchants are indexed $j$ and $j + 1$. Let $\alpha = \text{CTR}_{j+1,j+1}/\text{CTR}_{j,j+1}$ and let $\phi = \text{CTR}_{j,j}/\text{CTR}_{j,j+1}$. Now, consider the ranking function $R$ with $w_j = 1$, $w_{j+1} = \alpha$. All the other merchants are assigned a weight of 1. Suppose there exists a weighted VCG mechanism that always results in the same ranking as this ranking function. Let $\omega_i$ be the weight assigned by the VCG mechanism to merchant $i$, normalized such that $\omega_j = 1$. Then, the VCG mechanism chooses that ranking scheme that maximizes $\sum_{i=1}^{K} \omega_{m(i)}\text{CTR}(m(i), i)b_{m(i)}$, where $m(i)$ is the index of the merchant placed $i$ in the ranking scheme. Let $\rho$ be the ratio of the maximum click-through rate

to the minimum click-through rate over all merchants and positions, and let $\nu$ be the ratio of the maximum VCG weight to the minimum VCG weight. Also, let $\beta_{max} = \max\{1, 1/\alpha)$ and let $\beta_{min} = \min\{1, 1/\alpha)$. Consider the following set of bids: $b_i = (2n)^{j-i}\rho\nu\beta_{max}$ for $i = 1, \ldots, j-1$, $b_j = 1$, $b_{j+1} = 1/\alpha$, and $b_i = \beta_{min}/((2n)^{i-(j+1)}\rho\nu)$ for the rest. Then, it is easy to verify that merchant $i$ is placed at position $i$ for $i = 1, 2, \ldots, j-1$ and merchants $j$ and $j+1$ share the remaining two positions (i.e., positions $j$ and $j+1$) under both the ranking function $R$ as well as the VCG mechanism. The ranking score under $R$ of merchant $j$ and $j+1$ is exactly the same, namely 1. Therefore, the ranking function $R$ can be forced to place them in any chosen order by an infinitesimal change in the bids. In order for the VCG mechanism to produce the same ranking as $R$ after the change, VCG must rate both possible orderings of $j$ and $j+1$ equally as well, i.e. the weighted sum of utilities (with the above bids) must be the same for both possible orderings.

$$\phi + \omega_{j+1} = 1 + \omega_{j+1}\frac{\text{CTR}_{j+1,j}}{\text{CTR}_{j+1,j+1}} \tag{6.1}$$

We could also have set the bids of the other merchants such that they get ranks $1, \ldots, j$, leaving merchants $j$ and $j+1$ to compete for rank $j+1 = K$. Then, a reasoning similar to above would show that

$$1 = \omega_{j+1} \tag{6.2}$$

Putting equations 6.1 and 6.1 together, we get

$$\frac{\text{CTR}_{j+1,j}}{\text{CTR}_{j+1,j+1}} = \phi.$$

This completes the proof of the base case.

By the induction hypothesis, $\text{CTR}_{i,j}/\text{CTR}_{i,j+1} = \text{CTR}_{i',j}/\text{CTR}_{i',j+1}$ for all values of $i, i' \leq n$ and $\hat{j} < j \leq K-1$. Next consider $j = \hat{j}$. Consider the ranking function $R$ with $w_i = 1$ for all merchants $i$. Let $\omega_i$ be the weight assigned by the corresponding VCG mechanism to merchant $i$. Again consider a pair of merchants and re-index merchants such that the pair is indexed $j$ and $j+1$. Let $b_j = b_{j+1} = 1$. As before, we can set the bids of other merchants such that merchant $i$ is ranked $i$ for $i = 1, \ldots, j-1, j+2, \ldots, K$, while $j$

and $j+1$ share ranks $j$ and $j+1$. Since the ranking score given by $R$ is the same for both $j$ and $j+1$, the VCG mechanism must also be ambivalent towards their order, i.e.

$$\omega_j \text{CTR}_{j,j} b_j + \omega_{j+1} \text{CTR}_{j+1,j+1} b_{j+1} = \omega_j \text{CTR}_{j,j+1} b_j + \omega_{j+1} \text{CTR}_{j+1,j} b_{j+1}$$

which implies that

$$\frac{\omega_{j+1}}{\omega_j} = \frac{\text{CTR}_{j,j} - \text{CTR}_{j,j+1}}{\text{CTR}_{j+1,j} - \text{CTR}_{j+1,j+1}}. \tag{6.3}$$

We can also set the bids of the other merchants such that they get ranks $1, \ldots, j, j+3, \ldots, K$, leaving ranks $j+1$ and $j+2$ for merchants $j$ and $j+1$. Then, a reasoning similar to above would show that

$$\frac{\omega_{j+1}}{\omega_j} = \frac{\text{CTR}_{j,j+1} - \text{CTR}_{j,j+2}}{\text{CTR}_{j+1,j+1} - \text{CTR}_{j+1,j+2}} \tag{6.4}$$

From Equations 6.3 and 6.4, we get

$$\frac{\text{CTR}_{j,j} - \text{CTR}_{j,j+1}}{\text{CTR}_{j+1,j} - \text{CTR}_{j+1,j+1}} = \frac{\text{CTR}_{j,j+1} - \text{CTR}_{j,j+2}}{\text{CTR}_{j+1,j+1} - \text{CTR}_{j+1,j+2}}$$

Also, by induction hypothesis, we have

$$\frac{\text{CTR}_{j,j+1}}{\text{CTR}_{j,j+2}} = \frac{\text{CTR}_{j+1,j+1}}{\text{CTR}_{j+1,j+2}}$$

Using the above two equations and some elementary algebra, we get,

$$\frac{\text{CTR}_{j,j}}{\text{CTR}_{j+1,j}} = \frac{\text{CTR}_{j,j+1}}{\text{CTR}_{j+1,j+1}}$$

This completes the proof by induction. □

Although we have presented the theorem above for unbiased VCG, a similar statement holds for biased, weighted VCG is well. We can see this as follows. In each of the constraints 6.1, 6.2, 6.3 and 6.4 above, the two sides of the equation represent the rating given by the VCG mechanism to two different outcomes. If the biased VCG mechanism adds an unequal bias to the two outcomes, it can be viewed as adding a non-zero bid-independent

constant term to the right-hand side. The key idea is that we can scale up the bids uniformly without changing the ordering output by the ranking function $R$. Thus, the chosen VCG weights need to satisfy the constraints both for the scaled and the unscaled bid vector, which is impossible in the presence of a non-zero bid-independent constant term. Thus, the VCG mechanism must have added the same bias to both sides of each of the constraints, thereby leaving the constraints unchanged. We state the following theorem without proof.

**Theorem 6.2.2** *Let the number of merchants with non-zero click-through rates be $n > K$. If the click-through rates are not separable, then there exists a ranking function $R = (w_1, w_2, \ldots, w_n)$ for which there does not exist any set of weights for which biased, weighted VCG always yields the same ranking as the $R$.*

Interestingly, VCG is applicable under the separability assumption, with appropriately chosen weights. It is easy to verify the following theorem.

**Theorem 6.2.3** *Let the click-through rates be separable. Then the VCG mechanism having merchant $i$'s VCG weight set to $w_i/CTR_{i,1}$ always produces the same ordering as the ranking function $(w_1, \ldots, w_n)$.*

The above theorem implies that with the separability assumption, the ranking functions maximize a certain global utility function. In particular, the revenue-ranking scheme maximizes the total utility obtained by the merchants and the auctioneer.

## 6.3 The Truthful Auction

In this section, we will assume without loss of generality that the $i^{th}$ merchant also has the $i^{th}$ rank in the auction. The truthful auction is quite simple: For $1 \leq i \leq K$, set the price-per-click $p_i$ charged to merchant $i$ as:

$$\text{CTR}_{i,i} p_i = \sum_{j=i}^{K} (\text{CTR}_{i,j} - \text{CTR}_{i,j+1}) \frac{w_{j+1}}{w_i} b_{j+1}. \tag{6.5}$$

In other words,

1. For those clicks which merchant $i$ would have received at position $i+1$, she pays the same price as she would have paid at position $i+1$.

2. For the additional clicks, merchant $i$ pays an amount equal to the bid value of merchant $i+1$.

Since $w_i b_i \geq w_j b_j$ for $j > i$, it follows that $p_i \leq b_i$. Hence the price charged per click-through can be no larger than the submitted bid. We will refer to this auction as *Laddered_Auction*$(w_1, \ldots, w_n)$ or simply as the *laddered auction* when the $w_i$s are clear from the context.

## 6.4 Analysis

**Theorem 6.4.1** *Given fixed $w_1, \ldots, w_n$, the laddered auction is truthful. Further, it is the unique truthful auction that ranks according to decreasing $w_i b_i$.*

**Proof:** Consider a merchant $M$. Fix the bids of all the other merchants arbitrarily. With these bids, let $p(j)$ be the price charged by the laddered auction to merchant $M$ if her rank is $j$, with $p(K+1) = 0$. Note that the price charged depends only on merchant $M$'s rank and is independent of her exact bid value. Let $v_M$ be the true valuation of a single click for merchant $M$. If merchant $M$ bids $v_M$, let her be ranked $x$. Also, without loss of generality, assume that all the merchants are indexed such that merchant $j$ would be ranked $j$ if merchant $M$ bids $v_M$. Then, $w_j b_j \geq w_x v_x$ for all $j < x$ and $w_x v_x \geq w_j b_j$ for all $j > x$. To show that the auction is truthful, we will show that merchant $M$ cannot benefit by lying about her valuation. Among all ranks that give the merchant the highest profit (i.e., $\mathrm{utility} - \mathrm{price}$), let $r$ be the rank closest to $x$, i.e. the one with the least $|r-x|$. Now suppose that the merchant can benefit by lying, i.e., $r \neq x$. For a contradiction, we will show that there is a rank closer to $x$ which gives at least the same profit. For this, observe that if $r > x$, then the change in profit by moving to rank $r-1$ is $(\mathrm{CTR}_{x,r-1} - \mathrm{CTR}_{x,r})(v_x - \frac{w_r}{w_x} b_r)$, which is non-negative. On the other hand, if $r < x$, the change in profit in moving to rank $r+1$ is $(\mathrm{CTR}_{x,r+1} - \mathrm{CTR}_{x,r})(v_x - \frac{w_r}{w_x} b_r)$, which is again non-negative.

To show uniqueness, consider any truthful auction $\mathcal{A}$ that ranks the merchants in the decreasing order of $w_i b_i$. Consider any merchant $M$ and fix the bids of all the other merchants

arbitrarily. With these bids, let $p_{\mathcal{A}}(j)$ be the price charged by auction $\mathcal{A}$ to merchant $M$ if she is ranked $j$, with $p_{\mathcal{A}}(K+1) = 0$. Note that in a truthful auction, $p_{\mathcal{A}}(j)$ can depend on the bids of other merchants, but is independent of $M$'s bid (see Theorem 4.1.1). Assume, without loss of generality, that the other merchants are indexed such that merchant $i$ would be ranked $i$ if merchant $M$ bids $\infty$. To prove uniqueness, it suffices to show that for any truthful auction,

$$p_{\mathcal{A}}(j) - p_{\mathcal{A}}(j+1) = (\text{CTR}_{x,j} - \text{CTR}_{x,j+1})\frac{w_{j+1}}{w_x}b_{j+1} \tag{6.6}$$

First suppose that merchant $M$ has valuation $v_M = \frac{w_{j+1}}{w_x}b_{j+1} + \epsilon$. Then, if she bids truthfully, for sufficiently small $\epsilon > 0$, she is ranked $j$. The additional valuation per impression of being ranked $j$ instead of $j+1$ is given by $(\text{CTR}_{x,j} - \text{CTR}_{x,j+1})v_x$. Thus, this is the maximum amount that can be charged by a truthful auction for this additional valuation (otherwise, the merchant can benefit by bidding lower to get rank $j+1$). Since $\epsilon$ can be made arbitrarily small, this proves that

$$p_{\mathcal{A}}(j) - p_{\mathcal{A}}(j+1) \leq (\text{CTR}_{x,j} - \text{CTR}_{x,j+1})\frac{w_{j+1}}{w_x}b_{j+1} \tag{6.7}$$

Next, suppose that merchant $M$ has valuation $v_M = \frac{w_{j+1}}{w_x}b_{j+1} - \epsilon$. Then, if she bids truthfully, for sufficiently small $\epsilon > 0$, she is ranked $j+1$. The additional valuation per impression of being ranked $j$ instead of $j+1$ is given by $(\text{CTR}_{x,j} - \text{CTR}_{x,j+1})v_x$. Thus, this is the minimum amount that can be charged by a truthful auction for this additional valuation (otherwise, the merchant can benefit by bidding higher to get the $j^{th}$ rank). Since $\epsilon$ can be made arbitrarily small, this proves that

$$p_{\mathcal{A}}(j) - p_{\mathcal{A}}(j+1) \geq (\text{CTR}_{x,j} - \text{CTR}_{x,j+1})\frac{w_{j+1}}{w_x}b_{j+1}. \tag{6.8}$$

Putting together 6.7 and 6.8, we get 6.6, thereby completing the proof. □

**Corollary 6.4.2** *For any fixed $w_1, \ldots, w_n$, the laddered auction is the profit-maximizing truthful auction that ranks merchants by decreasing $w_i b_i$.*

## 6.5 Revenue Equivalence

In this section, we compare the revenue of the laddered auction to the revenue achieved by the next-price auctions currently being used. As mentioned earlier, truth-telling is not a dominant strategy for the existing auctions. Thus, we consider the revenue of the existing auctions under equilibrium conditions, i.e. a setting of bids for which no merchant can increase her profit by a unilateral change in his bid.

For separable click-through rates (see Definition 6.2), we show that there exists a *pure-strategy* Nash equilibrium under the next-price auction that yields the same revenue as the laddered auction. Let the weights used by the next-price auction be $(w_1, w_2, \ldots, w_n)$. Re-index the merchants in the decreasing order of $w_i v_i$ so that

$$w_i v_i \geq w_{i+1} v_{i+1} \quad \text{for } i = 1, \ldots n-1 \tag{6.9}$$

Let the click-through rates be separable with the click-through rate of merchant $i$ at position $j$ given by $\mu_i \theta_j$. Also let $\theta_{K+1} = 0$. Then, the bids $b_i$ for this Nash equilibrium are recursively defined by:

$$w_i b_i = \left(\frac{\theta_i}{\theta_{i-1}}\right) w_{i+1} b_{i+1} + \left(1 - \frac{\theta_i}{\theta_{i-1}}\right) w_i v_i \quad \text{for } i = K, \ldots, 1 \tag{6.10}$$

with the initialization $b_{K+1} = v_{K+1}$.

**Theorem 6.5.1 (Revenue-Equivalence Theorem)** *The bids defined by the recursive formula given in Equation 6.10 are in equilibrium. Moreover, the ranking induced by these bids is the same as the ranking induced by truthful bidding.*

**Proof:** To prove this, we unroll the recursion to get:

$$w_i b_i = \frac{1}{\theta_{i-1}} \sum_{j=i-1}^{K} (\theta_j - \theta_{j+1}) w_{j+1} v_{j+1}$$

Thus, $w_{i+1} b_{i+1}$ is a convex linear combination of $w_j v_j$ for $j = i+1, \ldots, K+1$. Since, $w_i v_i \geq w_j v_j$ for $j = i+1, \ldots, K+1$, we get $w_i v_i \geq w_{i+1} b_{i+1}$. We also know that $w_i b_i$ is a convex linear combination of $w_{i+1} b_{i+1}$ and $w_i v_i$. Hence, $w_i b_i \geq w_{i+1} b_{i+1}$. This shows that

the ranking induced by these bids is the same as that induced by truth-telling, i.e., merchant $i$ is ranked $i$ by the ranking function of the next-price auction.

Next, we will show that under the next-price auction, no merchant can gain by changing her bid unilaterally. Consider the merchant ranked (and indexed) $x$. With the above bids, she is making a profit of:

$$\begin{aligned} U(x) &= \mu_x \theta_x (v_x - b_{x+1}) \\ &= \mu_x \sum_{j=x}^{K} (\theta_j - \theta_{j+1}) \left( v_x - \frac{w_{j+1} v_{j+1}}{w_x} \right) \end{aligned}$$

If the merchant changes her bid in order to be ranked $y$, her profit becomes

$$\begin{aligned} U(x) &= \mu_x \theta_y (v_x - b_{y+1}) \\ &= \mu_x \sum_{j=y}^{K} (\theta_j - \theta_{j+1}) \left( v_x - \frac{w_{j+1} v_{j+1}}{w_x} \right) \end{aligned}$$

If the merchant decreases her bid in order to be ranked $y$, i.e. $y > x$, then the net change in profit is:

$$-\mu_x \sum_{j=x}^{y+1} (\theta_j - \theta_{j+1}) \left( v_x - \frac{w_{j+1} v_{j+1}}{w_x} \right)$$

By equation 6.9, $w_x v_x \geq w_{j+1} v_{j+1}$ for $j = x, \ldots, y+1$, which in turn implies that this change is non-positive. Similarly, if the merchant increases his bid in order to be ranked $y$, i.e. $y < x$, the net change in profit is:

$$\mu_x \sum_{j=y}^{x+1} (\theta_j - \theta_{j+1}) \left( v_x - \frac{w_{j+1} v_{j+1}}{w_x} \right)$$

Again, equation 6.9 implies that this change is non-positive. Thus, the bids are in equilibrium and none of the merchants can improve her profit by changing her bid unilaterally. □

Note that the merchants can achieve this equilibrium by solely using the knowledge of their true valuation and the current price being charged to them. To do this, the merchants start by bidding their true valuations, after which the $k^{th}$-ranked merchant changes her bid

to the one indicated in the formula above in order to prevent anybody from under-cutting her, followed by merchant $k-1$ changing her bid to the $b_{k-1}$ value defined above and so on. For example, consider four merchants $A$, $B$, $C$ and $D$ bidding for three slots. Let all four of them have a click-through rate of $0.5$ at the top slot, $0.4$ at the middle slot and $0.2$ at the bottom slot. Let the true valuations per click of the three merchants be 200, 150, 100 and 40 respectively. Let the ranking function be Google's revenue-ranking function. The merchants start off by bidding their true valuations, and $A$, $B$ and $C$ get the top, middle and bottom slot respectively, and make a profit of $25$, $20$ and $12$ respectively. At this point, merchant $B$ has an incentive to undercut $C$ by bidding $80$, which will result in $B$ making a profit of $22$, while $C$ makes a reduced profit of $8$. In order to remove any incentive for $B$ to undercut her, $C$ can change her bid to $70$ as prescribed by the above formula. At this point $B$ is making a profit of $32$. Now, $B$ faces the problem of $A$ trying to undercut her by bidding $100$ (say) in order to make a profit of $52$, reducing $B$'s profit to $25$. To remove $A$'s incentive to undercut her, $B$ can change her bid to $86$ as prescribed by the above formula.

With suitable assumptions, including separability of click-through rates, one can also use standard techniques such as the envelope theorem [SB94] to prove revenue equivalence. We omit the details since our first-principles analysis gives a stronger result in the form of a pure strategy Nash equilibrium under which the next-price auction is revenue-equivalent to the laddered auction, while the envelope theorem would only guarantee a mixed-strategy Nash equilibrium. Further, we obtain a simple and explicit characterization of the revenue-equivalent Nash equilibrium. Admittedly, these results show revenue equivalence to the next-price auction only. However, since the next-price auction is the auction currently in deployment, it is arguably the most interesting auction to consider in terms of showing revenue-equivalence.

### 6.5.1 Existence of Multiple Nash Equilibria

The foregoing discussion shows that there exists an equilibrium for the next-price auction which achieves the same ranking and the same revenue as the laddered auction. It should be pointed out that not all equilibria of the next-price auction have these properties. We next give an example that shows that there may exist other pure-strategy Nash equilibria

under which the next-price auction achieves a smaller revenue than the truthful auction, and yet others under which the next-price auction achieves a higher revenue.

Consider three merchants $A, B$ and $C$ having valuation-per-click of $500$, $480$ and $100$ respectively bidding for two slots. Assume that the click-through rates are separable, and that all the merchants have the same $\mu_i$ of 1, and that the position-specific factors are given by $\theta_1 = 0.2$ and $\theta_2 = 0.15$. Let the ranking function be revenue-ranking. Assuming that everyone follows the dominant strategy of truth-telling, the laddered auction earns a revenue of $15 + (15 + 24) = 54$. Moreover, if everyone bids truthfully, then the next-price auction would earn a revenue of $15 + 96 = 111$, more than twice the revenue of the laddered auction. However, truthful bidding is not an equilibrium for the next-price auction. One way to achieve equilibrium is for merchant A to change his bid to $110$ in which case the revenue earned is $15 + 22 = 37$. On the other hand, if equilibrium is achieved by merchant $B$ changing her bid from $480$ to $200$ before merchant $A$ changes his bid, a different equilibrium is reached. In this case, the revenue earned is $15 + 40 = 55$. In this particular example, unless merchant $B$ bids $200$ or lower, merchant $A$ will have an incentive to undercut her. This indicates that among all possible equilibria for this instance (excluding the ones where merchant $C$ bids more than her true valuation, as there is no incentive for merchant $C$ to do so), the highest revenue earned is $55$. In order to achieve an equilibrium that achieves the same revenue as the laddered auction, merchant $B$ could change her bid to $195$, again preventing under-cutting by merchant $A$.

## 6.6 A Merchant with a Budget Constraint

In this section, we consider a merchant who is interested in advertising on multiple keywords, and has a limited total budget of $B$ that she can spend on advertising on those keywords. For each of the keywords, we assume that the search engine publishes the current ranking score $w_j b_j$ for each slot $j$ corresponding to the keyword; here $w_j$ and $b_j$ refer to the weight and bid respectively of the merchant placed at the $j^{th}$ slot of the keyword. Given this knowledge and her own weight $w_M$, the merchant can compute how much she needs to bid (and pay) in order to be placed at a certain slot for any given keyword, i.e., effectively she knows the price $p_{jk}$ of the $j^{th}$ slot of the $k^{th}$ keyword she is interested in.

In addition, she knows her valuation $v_{jk}$ of a click she receives when placed on the $j^{th}$ slot of the $k^{th}$ keyword (this valuation may or may not vary with slot). Now she is faced with the optimization problem of deciding how much money to spend on advertising on each keyword. If each keyword was searched for infinitely often during the course of a day, it is in her best interest to spend all her budget on the slot-keyword pair with the maximum valuation-per-unit-price. However, if the number of queries for a given keyword are finite, the merchant might not be able to spend all her budget on a single keyword. Let us assume that she knows the number of times a keyword is queried over the course of a day. Now the problem is to find the optimal way to split the available budget $B$ among the keyword-slot pairs with the goal of maximizing valuation.

We can compute the expected price-per-impression $\pi_{jk}$ of a given keyword-slot pair $(j, k)$ by using the price-per-click and the merchant's click-through rate for the keyword-slot pair. Similarly, assuming the merchant is risk-neutral, we can compute her valuation of an impression $\nu_{jk}$ for a given keyword-slot pair $(j, k)$ by using her valuation-per-click and her click-through rate for the keyword-slot pair.

**A Greedy Strategy.** One simple strategy is buy impressions greedily according to decreasing valuation-per-unit-cost ratio. However, a naïve implementation would end up buying multiple slots on a single impression of a keyword. To prevent this, we devise a slightly more clever greedy algorithm as follows. For each keyword, we have a set of valuation-per-unit-cost ratios corresponding to different slots on the page. Among all the available keyword-slot pairs, we pick the pair with the maximum valuation-per-unit-cost ratio and buy all available impressions of the keyword (or as much as we can with the current remaining budget). We now observe that replacing one impression at the just-bought slot $j$ of keyword $k$ with an impression at another slot $j'$ (with a higher valuation-per-impression) of the same keyword $k$ incurs an additional cost of $\hat{\pi}_{jk'} = \pi_{jk'} - \pi_{jk}$ and gets an additional valuation of $\hat{\nu}_{jk'} = \nu_{jk'} - \nu_{jk}$. Thus, for the remaining slots on the page, we update the valuation-per-unit-cost to the ratio to $\hat{\nu}_{jk'}/\hat{\pi}_{jk'}$. Now we repeat till the remaining budget is no longer sufficient to buy any more impressions of the current best keyword-slot pair.

**Performance.** Until the final step, we always buy the slot that gives the most valuation for money. The only possible reason for the solution being sub-optimal is the unspent budget. The potential loss due to this is no more than the valuation of an impression on the best slot at the end of the algorithm.

**Theorem 6.6.1** *Let the maximum valuation of a single impression at any slot of any keyword be $q$, and let $OPT$ be the total payoff (valuation) achieved by the optimal algorithm. Then, the greedy algorithm above achieves a payoff of at least $OPT - q$.*

Note that valuation from a single impression is usually quite small in practice. This algorithm gives similar performance guarantees whenever the valuation increases linearly with the cost as we buy more and more impressions of a slot on a page.

**NP-hardness.** The above problem can be shown to be NP-hard by a straightforward reduction from the knapsack problem: for each item, create a page with a single impression, with the cost of that impression (i.e. the cost of a click multiplied by the click-through rate) equal to the size of the item and the valuation equal to the profit associated with the item. Let the budget be equal to the size of the knapsack. Then the problem of maximizing the total valuation is equivalent to the problem of maximizing profit for the knapsack problem.

We note that the advertising problem instance created in the above reduction is not a very realistic one, since in most real-life scenarios, the number of available impressions of a single page is quite large.

# Chapter 7

# Conclusions

We have presented solutions for several problems that have arisen due to the pervasive use of the Internet and the networking infrastructure in general. The set of problems considered fall into two broad areas: (a) protection of data privacy, and (b) selling advertisement space on the Internet.

With respect to data privacy, we first considered the problem of **anonymizing databases** before dissemination, in order to safeguard the privacy of the individuals described by the databases. For the privacy framework of $k$-Anonymity, we presented approximation algorithms that anonymize databases while maximizing the utility of the anonymized database. Then, we studied the problem of enabling two or more parties to **securely and efficiently compute the $k^{th}$-ranked element** of a set split between them, without revealing any information not implied by the value of the output. We presented protocols with polylogarithmic overhead for this purpose, improving upon the linear overhead of earlier solutions.

Next, we considered the problem of selling advertisements space on web pages. We observed that the use of a truthful selling mechanism would considerably simplify the task of bidding, potentially attracting more advertisers to the online advertising market. We presented truthful auctions for two problem formulations in this setting — one that models selling a single slot on a web page with known budget constraints, and another that models selling multiple slots on a web page with no budget constraints.

Several problems still remain open. We mention a couple of them here:

**Extensions of the $k$-Anonymity framework.** One source of concern about the privacy guarantees under the $k$-Anonymity model is that for a given record in the public database, all the $k$ records corresponding to it in the anonymized table might have the same value of the sensitive attribute(s) (the attribute *Diseases* in our examples), thus revealing the sensitive attribute(s) conclusively. To address this issue, we could add a constraint that specifies that for each cluster in the $k$-anonymized table, the sensitive attribute(s) should take at least $r$ distinct values. This would be an interesting extension of the basic $k$-Anonymity framework. Another interesting direction of research is to extend the model to deal with changes in the database. A hospital may want to periodically release an anonymized version of its patient database. However, releasing several anonymized versions of a database might leak enough information to enable *record linkages* for some of the records. It would be useful to extend the $k$-ANONYMITY framework to handle inserts, deletes and updates to a database.

**Selling multiple advertisement slots with budget constraints.** A big open problem in the area of auction design for online advertising is the problem of designing an auction that takes into account both the interaction between multiple slots on a web page as well as budgets constraints for advertisers. An interesting extension would be to design auctions for selling a collection of keywords, when each advertiser is interested in a subset of the keywords and would like to spend her budget in a way that maximizes her combined utility.

# Bibliography


[AA01] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 247–255, 2001.

[ACCK01] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic security for mobile code. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 2–11, 2001.

[AFK+05] G. Aggarwal, T. Fèder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proceedings of the 10th International Conference on Database Theory*, pages 246–258, January 2005.

[AH06] G. Aggarwal and J. Hartline. Knapsack auctions. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[AMP04] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the $k^{th}$-ranked element. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 44–55, 2004.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 439–450, May 2000.

[AST05] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 251–262, 2005.

[AT01] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, page 482, 2001.

[BCI+05] C. Borgs, J. Chayes, N. Immorlica, M. Mahdian, and A. Saberi. Multi-unit auctions with budget-constrained bidders. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 44–51, 2005.

[BH05] A. Blum and J. Hartline. Near-optimal online auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1156–1163, 2005.

[BMR90] D. Beaver, S. Micali, , and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 503–513, 1990.

[Can00] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[CDM+05] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Proceedings of the First Theory of Cryptography Conference*, pages 363–385, 2005.

[CIK+01] R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R. Wright. Selective private function evaluation with applications to private statistics. In *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing*, pages 293–304, 2001.

[Cla71] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 494–503, 2002.

[CMS99] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 402–414, 1999.

[Cor88] G.P. Cornuejols. General factors of graphs. *Journal of Combinatorial Theory*, 45:185–198, 1988.

[Cra00] R. Cramer. Introduction to secure computation, 2000. Available at `http://www.brics.dk/~cramer/papers/CRAMER_revised.ps`.

[DHM79] P. Dasgupta, P. Hammond, and E. Maskin. The implementation of social choice rules: Some results on incentive compatibility. *Review of Economic Studies*, 46:185–216, 1979.

[DN03] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210, 2003.

[DN04a] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in Cryptology: Proceedings of Crypto*, pages 528–544, 2004.

[DN04b] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in Cryptology: Proceedings of Crypto*, pages 528–544, 2004.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28:637–647, 1985.

[EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 211–222, June 2003.

[FGHK02] A. Fiat, A. Goldberg, J. Hartline, and A. Karlin. Competitive generalized auctions. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 72–81, 2002.

[FIM+01] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. Secure multiparty computation of approximations. In *Proceedings of the 28th International Conference on Algorithms and Logic Programming*, pages 927–938, 2001.

[Fis01] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Proceedings of CT-RSA 2001: The Cryptographers' Track at RSA Conference*, pages 457–472, 2001.

[FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 1–19, 2004.

[For05] US Online marketing forecast: 2005 to 2010. A Forrester Inc. report, May 2005.

[FY92] M. Franklin and M. Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 699–710, 1992.

[GH03a] A. Goldberg and J. Hartline. Competitiveness via consensus. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 215–220, 2003.

[GH03b] A. Goldberg and J. Hartline. Envy-free auctions for digital goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 29–35, 2003.

[GH05] A. Goldberg and J. Hartline. Collusion-resistant mechanisms for single-parameter agents. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 620–629, 2005.

[GHK+02] A. Goldberg, J. Hartline, A. Karlin, M. Saks, and A. Wright. Competitive auctions and digital goods. *Games and Economic Behavior*, 2002. Submitted for publication. An earlier version available as InterTrust Technical Report from `http://www.star-lab.com/tr/tr-99-01.html`.

[GHK+05] V. Guruswami, J. Hartline, A. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1164–1173, 2005.

[GHKS04] A. Goldberg, J. Hartline, A. Karlin, and M. Saks. A lower bound on the competitive ratio of truthful auctions. In *21st Annual Symposium on Theoretical Aspects of Computer Science*, pages 644–655, 2004.

[GHW01] A. Goldberg, J. Hartline, and A. Wright. Competitive auctions and digital goods. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 735–744, 2001.

[GL77] J. Green and J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[GMP97] P. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *Proceedings of the 23rd International Conference Very Large Data Bases*, pages 466–475, 1997.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[Gol98] O. Goldreich. Secure multi-party computation. In *Theory of Cryptography Library*, 1998.

[Goo05] Quarterly report filed by Google Inc. with the SEC for the period ending March 31, 2005.

[Gro73] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[HM05] J. Hartline and R. McGrew. From optimal limited to unlimited supply auctions. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 175–182, 2005.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology: Proceedings of Crypto*, pages 145–161, 2003.

[JKM+98] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *Proceedings of the 24th International Conference on Very Large Data Bases*, pages 275–286, 1998.

[Kan94] V. Kann. Maximum bounded H-matching is MAX SNP-complete. *Information Processing Letters*, 49:309–318, 1994.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 20–31, 1988.

[KMN05] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 118–127, June 2005.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KPR03] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. *Journal of Computer and System Sciences*, 6:244–253, 2003.

[LOS99] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 96–102, 1999.

[LP02] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[MCWG95] A. Mas-Collel, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[MN02] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, pages 379–384, 2002.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MSVV05] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. In *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science*, 2005.

[MW04] A. Meyerson and R. Williams. On the complexity of optimal k-Anonymity. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 223–228, June 2004.

[Mye79] R. Myerson. Incentive compatibility and the bargaining problem. *Econometrica*, 47(1):61–73, 1979.

[NN01] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 590–599, 2001.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, 2001.

[NR99] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 129–140, 1999.

[PGI99] V. Poosala, V. Ganti, and Y. Ioannidis. Approximate query answering using histograms. *IEEE Data Engineering Bulletin*, 22(4):5–14, 1999.

[Pin03] B. Pinkas. Cryptographic techniques for privacy-preserving data mining, 2003.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Technical report, Aiken Computation Laboratory, 1981.

[Rob79] K. Roberts. The characterization of implementable choice rules. In J. Laffont, editor, *Aggregation and Revelation of Preferences*, pages 321–348. North Holland Publishing Company, 1979.

[Rod82] M. Rodeh. Finding the median distributively. *Journal of Computer and Systems Sciences*, 24:162–166, 1982.

[Sam01] P. Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[Sav72] J. Savage. Computation work and time on finite machines. *Journal of the ACM*, 19(4):660–674, 1972.

[SB94] C.P. Simon and L. Blume. *Mathematics for Economics*. W.W. Norton & Company, Inc., New York, 1994.

[SS98] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of the 17th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, page 188, 1998.

[Swe00] L. Sweeney. Uniqueness of simple demographics in the U.S. population. Technical Report LIDAP-WP4, Laboratory for International Data Privacy, Carnegie Mellon University, Pittsburgh, PA, 2000.

[Swe02] L. Sweeney. k-Anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[Vic61] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[Yah05] Quarterly report filed by Yahoo! Inc. with the SEC for the period ending March 31, 2005.

[Yao86] A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.