SOME APPROXIMATION ALGORITHMS FOR MULTI-AGENT SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Lei Wang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Algorithm,Combinatorics and Optimization, College of Computing

Georgia Institute of Technology
August 2011

SOME APPROXIMATION ALGORITHMS FOR MULTI-AGENT SYSTEMS

Approved by:

Professor Robin Thomas,
Committee Chair
Algorithm,Combinatorics and
Optimization, College of Computing
Georgia Institute of Technology

Professor Vijay Vazirani, Advisor
Algorithm,Combinatorics and
Optimization, College of Computing
Georgia Institute of Technology

Professor Vijay Vazirani
College of Computing
Georgia Institute of Technology

Professor Nina Balcan
College of Computing
Georgia Institute of Technology

Professor Prasad Tetali
College of Computing
Georgia Institute of Technology

Professor Robin Thomas
School of Mathematics
Georgia Institute of Technology

Professor William Cook
School of ISyE
Georgia Institute of Technology

Date Approved: ——

I delicate this thesis to my parents,

Xiaodong Wang and Jun Tian Wang,

for their selfless love.

ACKNOWLEDGEMENTS

First of all, I owe my deepest gratitude to my advisor Vijay Vazirani who gave me the invaluable guidance throughout my PhD study. Vijay is a very good advisor to me in many aspects: he introduced me into the palace of theoretical computer science, gave me freedom in choosing research problems and provided me good career advice. I enjoy all the meetings and conversations we had. At last, I will always remember the warm encouragement from Vijay when my paper was rejected from STOC'11.

Secondly, I am indebted to Gagan Goel from whom I learned a lot during the years when we worked together. As a senior student, he showed me how to find my own research problems and helped me improve my writing. I am also grateful to Changyuan Yu who introduced me to the area of mechanism design and taught me how to give a talk.

In addition, I would like to thank my other co-authors: Nina Balcan, Deeparnab Chakrabarty, Xue Chen, Florin Constantin, Zhiyi Huang, Chinmay Karande, Pinyan Lu, Pushkar Tripathi and Yuan Zhou. I won't forget those days when we shared ideas together. Special thanks to my thesis reader Nina Balcan who supported me for summer 2011 when I was writing this thesis and special thanks to Pinyan Lu who hosted me as an intern at Microsoft Research in the summer of 2009.

Thank my parents Xiaodong Wang and Jun-Tian Wang who are always there when I need them. Also, very importantly, I owe a great gratitude to my girlfriend Peng Tang for her companion and support all the time. Thank all my friends in Atlanta who made my PhD life cheerful.

At last, I would like to thank my thesis committee Nina Balcan, William Cook, Prasad Tetali, Robin Thomas and Vijay Vazirani for their patient and time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis makes a number of contributions to the theory of approximation algorithm design for multi-agent systems. In particular, we focus on two research directions. The first direction is to generalize the classical framework of combinatorial optimization to the submodular setting, where we assume that each agent has a submodular cost function. We show hardness results from both the information-theoretic and computational aspects for several fundamental optimization problems in the submodular setting, and provide matching approximation algorithms for most of them. The second direction is to introduce game-theoretic issues to approximation algorithm design. Towards this direction, we study the application of approximation algorithms in the theory of truthful mechanism design. We study both the standard objectives of revenue and social welfare, by designing efficient algorithms that satisfy the requirement of truthfulness and guarantee approximate optimality.

CHAPTER I

INTRODUCTION

Most natural optimization problems, including those that arise in important applications, are computationally hard, and understanding the approximabilities of these problems becomes a compelling subject in both mathematics and theoretical computer science. Over the last two decades, the study of approximation algorithms has blossomed into a rich field ([51]).

Recently, rapidly developing technology such as the Internet has provided a large scale computing platform. As a consequence, a feature that arises in many computational problems is the presence of multiple agents that can interact with each other. This has engendered a rich set of economic and computational challenges in which approximation serves as a guiding principle, resulting in two new directions in the area of approximation algorithm design.

The first direction is a rectification of the classical framework of combinatorial optimization to fit in the multi-agent setting. Usually in a multi-agent system, each agent has a distinct valuation function representing her own interest. As a result, this brings an additional task of allocating the object among agents into the optimization algorithm design. At the same time, linear functions do not always model the complex dependencies of the agents' valuations. For example, one might give a discount to her cost function in order to compete with the other agents. These features of multi-agent system introduce new issues into combinatorial optimization and resolving them requires new computational models and algorithmic techniques.

The second direction is the introduction of game-theoretic considerations in approximation algorithm design. In the traditional model, an algorithm is not supposed

to question the origin of the input but is supposed to perform the optimization on the given input. However, in the new model of computation, we assume that different parts of the input originate from different self-interested agents, each of which has her own optimizations to perform. As a consequence, the agent may misreport the true value of her input to manipulate the algorithm. This usually leads to a loss in the performance. Therefore, a desired property of an algorithm is to motivate the agents to report their true inputs, so that the algorithm gives a globally optimized output.

1.1 Contribution

In this thesis, we contribute to both directions mentioned above.

Towards the first direction, we propose and study the model of *Submodular Combinatorial Optimization*. In our model, we are given a set of elements and a collection of combinatorial objects defined over subsets of elements (for instance, spanning trees defined over subsets of edges in a graph). We are also given a set of agents, each of which specifies a submodular cost function. Our goal is to find a combinatorial object and an allocation of its elements to the agents, such that the total cost is minimized.

We will study various fundamental optimization problems in our model including reverse auction, vertex cover, spanning trees, perfect matching and shortest path. From a practical viewpoint, each of these problems is meaningful in its own right. For example, shortest path and spanning trees are used in network design problems, and it is natural to assume that different agents could have different submodular cost functions depending on the set of edges they can construct cheaply. From a theoretical perspective, one would like to extend the tools and techniques developed in classical approximation algorithm design to as general a setting as possible. Submodular function is a natural generalization where one would expect to be able to extend the techniques.

In the second direction, we will explore the application of approximation algorithms in truthful mechanism design. A typical goal in mechanism design is to optimize the *social welfare*. It is well known that the *VCG* mechanism ([10, 22, 52]) which maximizes the social welfare is truthful. However, it suffers from many weaknesses that are frequently decisive ([2]). A major concern is that VCG is not computationally efficient in general and this is exactly the place where approximation algorithms come in. Unfortunately, the simple combination of approximation and VCG is not guaranteed to be truthful. This raises the important open question (see [39]) of designing approximation algorithms that preserve truthfulness. In this thesis, we will resolve the question completely for the broad class of *symmetric single-parameter* problems.

Another standard objective in mechanism design is *revenue optimization*. Myerson initiated this study in his seminal paper [38] by characterizing revenue-optimal mechanisms for the *single-item auction* with independently distributed agents. In the case of correlated agents, the optimal auction is still not well understood. An economic approach for solving the problem is to generalize Myerson's characterization. Unfortunately, most results obtained via this approach are for restricted special cases. In this thesis, we will study the problem of single-item auction with correlated agents from a computer science aspect. In other words, instead of providing a characterization of the optimal auction, we will design efficient truthful mechanisms that guarantee approximately optimal revenue.

1.2 Organization and Credits

To present our results, we organize the thesis as follows:

Results in submodular combinatorial optimization will be presented in chapters 2 and 3.

- **Chapter 2**. In this chapter, we will study the submodular combinatorial optimization problems in the *oracle model*, in which we assume that every submodular function is given by a value oracle. We will study five fundamental combinatorial optimization problems in this setting: reverse auction, vertex cover, spanning tree, shortest path and perfect matching. We start with the single-agent setting and generalize the results to the multi-agent case. We provide approximation algorithms and matching information-theoretic lower bounds for these problems.

- **Chapter 3**. As a follow up work to chapter 2, we will study a class of succinctly representable submodular functions called discounted price functions. This class function has its motivations from both theory and practice. We will design approximation algorithms for the five problems studied in chapter 2 and obtain computational hardness results.

Next, we will study the application of approximation algorithms in mechanism design in chapters 4 and 5.

- **Chapter 4**. A single-parameter mechanism design problem is symmetric, if each agent's valuation is represented by a single value and the alternative space is closed under permutation. In this chapter, we consider the class of symmetric single-parameter problems and derive a black-box that converts any approximation algorithm to a truthful mechanism with essentially the same approximation factor. A central question in algorithmic mechanism design is whether truthfulness introduces additional burden in approximation algorithm design, and our result implies that for the class of symmetric single-parameter problems, mechanism design is as easy as algorithm design.

- **Chapter 5**. We consider the revenue maximization aspect of mechanism design in this chapter. In particular, we will study the k-lookahead mechanism, which is

class of truthful mechanisms for the single-item auction with correlated agents. Briefly speaking, it selects the k highest bidders and write a linear program that captures the optimal auction among these k-bidders to output the allocation. One can show that as long as k is a constant, the linear program has polynomial size, hence the mechanism is efficient. By a delicate analysis, we show that the approximation ratio of k-lookahead mechanism is at least $\frac{e^{1-1/k}}{1+e^{1-1/k}}$. Furthermore, we prove that our analysis is tight for 2-lookahead.

CHAPTER II

SUBMODULAR COMBINATORIAL OPTIMIZATION: ORACLE MODEL

2.1 Introduction

A multitude of fundamental computational problems with real-world applications can be cast in the following framework: We are given a set X of elements, a collection C of subsets of X (i.e. $C \subseteq 2^X$) and a cost function f over the subsets of X. The collection C is typically specified via a combinatorial structure like a matroid or a graph property (for instance, the set of all spanning trees in a graph). The objective is to select a set $S \in C$ that minimizes $f(S)$.

A major focus in theoretical computer science has been on linear cost functions. The study of combinatorial problems with linear cost functions has led to great developments in the theory of exact and approximation algorithms. However, linear cost functions do not always model the complex dependencies of the costs in a real-world scenario. Often, they only serve as an approximation to the original functions. As a result, even though we might have a good algorithm for solving some linear optimization problem, the output solution can still be suboptimal.

Another feature that arises in practice is the presence of multiple agents, where each agent has her own cost function. Thus, in the optimal solution, each agent might build only a part of the required combinatorial structure. For example, the Internet is a complex multi-agent system where each service provider owns only a part of the network. For linear cost functions, it is easy to see that having multiple agents doesn't change the complexity of the original problem. However, this is not the case for more general cost functions.

Motivated by these considerations, we define the following class of *submodular combinatorial optimization* problems - We are given a set of elements X and a collection $C \subseteq 2^X$. We are also given m agents, where each agent i specifies a normalized monotone submodular cost function $f_i : 2^X \rightarrow R^+$. The goal is to find a set $S \in C$ and a partition $S_1, ..., S_m$ of S such that $\sum_i f_i(S_i)$ is minimized.

Submodular functions form a rich class and capture the natural properties of economies of scale and the law of diminishing returns . A function $f : 2^X \rightarrow R^+$ is said to be submodular iff for any two sets S and $T \subseteq X$, $f(S)+f(T) \geq f(S \cup T)+f(S \cap T)$. Function f is said to be monotone if $f(S) \leq f(T)$ for any $S \subseteq T$, and normalized if $f(\emptyset) = 0$. Note that linear functions, i.e. functions of the form $f(S) = \sum_{i \in S} a_i$, are a special and classically studied subcase of submodular functions, Since a submodular function is defined over an exponentially large domain, we will work with the *value oracle* model in which an oracle will return the value of $f(S)$, when queried with the set $S \subseteq X$.

Notice that by fixing the collection C to any particular combinatorial structure, one can define a subclass of the problems of interest. In this chapter, we study the following fundamental problems in our setting :

- **Combinatorial Reverse Auction**: We are given a set X of elements and the collection C consists of only the set X, i.e., in the required solution all the elements must be covered. This models the situation where a set of jobs needs to be assigned to multiple workers.

- **Submodular Vertex Cover**: We are given an undirected graph $G(V, E)$. Element set X is the same as the set of vertices V and the collection C consists of all the vertex covers of the graph. Recall that a set $S \subseteq V$ is a vertex cover if every $e \in E$ is incident on a vertex in S.

- **Submodular Shortest Path**: We are given a connected undirected graph

$G(V, E)$, and a pair of vertices $s, t \in V$. Element set X is the same as the set of edges E and the collection C consists of all the paths from s to t.

- **Submodular Minimum Perfect Matchings**: We have a undirected graph $G(V, E)$ that contains at least one perfect matching. Element set X is the set of all edges, and the collection C is defined as the set of all perfect matchings of G.

- **Submodular Minimum Spanning Tree**: We are given a connected undirected graph $G(V, E)$. Element set X is the set of all edges, and the collection C is the set of spanning trees of G.

For each of the above problems, we study both the single agent and the multi-agent setting.

2.1.1 Our Results

We give an approximation algorithm and a matching information theoretic lower bound for each of the problems that we mentioned earlier[1]. In case of shortest path, minimum spanning tree and minimum perfect matching problems, the bounds established are polynomial and tight up to poly-logarithmic factors. Ignoring these logarithmic factors, we present these results in the table 1. For the reverse auction problem, m is the number of agents and n is the number of items, whereas for all other problems, n is the number of vertices in the instance graph.

Note that the minimum perfect matching and minimum spanning tree problems, which are polynomial time solvable with linear cost functions, have a large hardness factor with submodular cost functions. We would like to draw attention to our lower bound result for the vertex cover problem in the single agent case. In the classical vertex cover problem, the best known approximation factor is 2, and the best known

[1]With the exception of the multi-agent submodular shortest path problem. We comment on this aberration in Section 2.5.2.

Table 1: Results of Submodular Combinatorial Optimization

	Single-agent		Multi-agent	
	Lower bound	Upper bound	Lower bound	Upper bound
Reverse Auction	1	1	$\Omega(\log n)$	$\min(m, \log n)$ [25]
Vertex Cover	$2-\epsilon$	2	$\Omega(\log n)$	$2\log n$
Shortest Path	$\Omega(n^{2/3})$	$O(n^{2/3})$	$\Omega(n^{2/3})$	$O(n)$
Perfect Matching	$\Omega(n)$	n	$\Omega(n)$	n
Spanning Tree	$\Omega(n)$	n	$\Omega(n)$	n

hardness of approximation is 1.3606 (assuming $P \neq NP$) [12]. Khot et al. [34] showed that achieving a factor of $2-\epsilon$ 'might be' hard by presenting a hardness result based on UGC conjecture [32]. Our results for the single agent submodular vertex cover problem implies that, if the cost function over the set of vertices is submodular, then the optimal approximation factor is indeed 2.

Our hardness results use information theoretic arguments and follow the framework explained in Section 2.2, with some modifications specific to each problem. Our algorithms are based on LP rounding or greedy methods.

We would like to point out that our results for perfect matchings and spanning trees extend to the class of subadditive cost functions, and to related combinatorial structures such as Steiner trees.

2.1.2 Related Work

Submodular functions have been of great interest in optimization in the past. The most fundamental optimization problem concerning submodular functions is, perhaps, the non-monotone submodular function minimization problem. A sequence of papers in this direction [17, 27, 28, 30, 40, 46] has resulted in fast strongly polynomial time combinatorial algorithms. Another related work is that of non-monotone submodular function maximization [15].

Another body of work in optimization over submodular functions deals with welfare maximization [9, 16, 33, 53]. In this context, the reverse auction problem that we study, can be thought of as submodular welfare minimization. Calinescu et al. [9] studied submodular function maximization subject to matroid constraints. They showed that their problem contains as subcases, many other allocation problems, thus giving a unified framework for studying such problems. Matching information theoretic lower bounds were established in [37].

Very recently, Svitkina and Fleischer [49] studied submodular objective function for problems like sparsest cut, load balancing, and knapsack. They gave $O\left(\sqrt{\frac{n}{\log n}}\right)$ upper and lower bounds for all these problems, showing that all these problems become much harder under submodular costs. For the submodular reverse auctions, where a set of n goods has to be allocated to m agents (i.e. collection set $C = \{X\}$) with submodular cost functions to minimize the overall cost, a simple greedy algorithm is known to have a factor $\log n$ [25]. Goemans et al. [21] gave an algorithm for constructing explicit approximate submodular functions by querying polynomial number of times to the original submodular function. Some other related work in optimization that uses submodular functions include [25, 47, 48, 50, 54].

Recall that the problems we study in this paper are very well studied in presence of linear cost functions. Shortest path, perfect matching and spanning tree can be solved exactly in polynomial time. Algorithm for vertex cover with factor 2 for weighted graphs was first given by [5]. The best known hardness of approximation for Vertex Cover is 1.3606 (assuming $P \neq NP$) [12]. Using UGC conjecture [32], Khot and Regev [34] showed that achieving a factor of $2 - \epsilon$ is hard.

2.2 Preliminaries: Information Theoretic Lower Bounds

A problem is said to have information theoretic lower bound of α if any randomized algorithm that approximates the optimum to a factor α with high probability requires

super-polynomial number of queries to the value oracle.

By Yao's principle, it suffices to establish the lower bounds for deterministic algorithms acting on an input which is picked randomly from some fixed distribution. To show these approximation gaps, we follow the general framework which was also used in [15, 21, 49]. We will outline this framework in the single agent setting.

The idea is to first choose a problem instance which has a suitably large collection set $C \subseteq 2^X$ of interest. For example, for the spanning tree problem, we choose a graph that has exponentially many spanning trees. Then we design two submodular cost functions f and g. Typically, g is deterministically picked, whereas f is chosen from a distribution. The choice of f and g relies on the following two properties: a) The optimum values of f and g over C must differ by a large factor, and b) f and g must be 'hard to distinguish' in the sense that, for any deterministic query $Q \subseteq X$, both functions return the same value with a high probability.

By the union bound and a computation path argument [15, 49], a deterministic algorithm making polynomially many queries cannot distinguish between f and g. Combining this with the gap in the optima of f and g, one proves the lower bound.

2.3 Combinatorial Reverse Auction

In this problem we are given a set J, of n elements and m agents. For each agent i we have a normalized monotone submodular cost function $f_i : 2^J \to \mathbb{R}^+$. We wish to partition the elements among the agents to minimize the total cost. We prove a $\Omega(\log n)$ information theoretic hardness result and provide an algorithm that matches this bound. We also prove the same algorithm to be m-approximate. Another $\log n$-approximate algorithm for this problem had previously appeared in [25].

2.3.1 Proof of hardness

As discussed in Section 2.2, the idea is to construct a deterministic instance and a random instance of the combinatorial reverse auction so that the optimal solution of

these two instances differ by a factor of $\Omega(\log n)$, and then show that with high probability, a deterministic algorithm which uses only polynomially many value queries can not distinguish between these two instances.

The deterministic instance we will use is the following: There are m agents and a set J of $n = m(m+1)^2/4$ elements. The elements are equally partitioned into m blocks $J_1, J_2, ..., J_m$. We will choose m such that $m = 2^d - 1$, for some d. Now each number i between 1 and m can be represented as a vector $\mathbf{a}_i$ in $GF[2]^d$. Let $G_i = \bigcup_{1 \leq k \leq m,\ \mathbf{a}_i \cdot \mathbf{a}_k = 1} J_k$. For each i, $1 \leq i \leq m$, agent i is only interested in elements in G_i. It is easy to see that G_i consists of exactly $\frac{m+1}{2}$ blocks and for each block there are $(m+1)/2$ agents who are interested in it. Now, we define the cost function $g_i : 2^J \longrightarrow \mathbb{R}^+$ as follows:

$$g_i(S) = \begin{cases} \min\{|S|, (m+1)^2/4\} & \text{If } S \subseteq G_i \\ \infty & \text{Otherwise} \end{cases}$$

Let us analyze the optimal cost of this instance. We say that an agent is *marked* if the total size of elements assigned to him is at least $(m+1)^2/4$. Among all the optimal solutions, let OPT be the one that maximizes the number of marked agents. We claim that at least d agents are marked in OPT. Suppose not, then without loss of generality, we may assume $M = \{1, 2, ..., t\}$ to be the set of marked agents and $t < d$. The system of linear equations $\mathbf{a}_i \cdot \mathbf{x} = 0, \forall 1 \leq i \leq t$ has at least one solution $\mathbf{x}^* \in GF[2]^d$, since number of equations is less than the number of variables. Let k be the number between 1 and m corresponding to the vector $\mathbf{x}^*$. This implies that no agent in M is interested in block J_k. Let $A_k = \{i_1, i_2, ..., i_w\}$ be the set of agents who are assigned elements from J_k. Then $A_k \cap M = \emptyset$. Therefore, we can mark one more agent by transferring the elements in J_k from agents $i_2, ..., i_w$ to agent i_1 without changing the cost of the new solution. This is a contradiction because of the choice of OPT. Hence, the optimal cost of this instance is at least $(m+1)^2 d/4$.

Next we will describe the randomized instance which has same the set of agents and elements as the deterministic instance. Also, each agent is interested in the same set of elements. However, the cost function for each agent is picked in a randomized manner. We describe in detail below.

For each element, assign it uniformly at random to one of the agents who is interested in it. Let S_i be the set of elements which agent i gets. Clearly $(S_1, S_2, ..., S_m)$ forms a partition of the element set J. We define the cost function $f_i : 2^J \longrightarrow \mathbb{R}^+$, for agent i as follows: if $S \subseteq G_i$,

$$f_i(S) = \min\left(\ |S \cap \overline{S_i}| + \min\left\{|S \cap S_i|, (1+\delta)(m+1)/2 \ \right\}, \ (m+1)^2/4\right),$$

and otherwise $f_i(S) = \infty$. Here $\delta > 0$ is a fixed constant.

Now we show that with high probability, a deterministic algorithm using only polynomially many value queries can not distinguish between $\mathbf{f} = (f_1, f_2, ..., f_m)$ and $\mathbf{g} = (g_1, g_2, ..., g_m)$. We prove the following lemma.

Lemma 2.3.1 *For any subset S of elements and any i, $1 \leq i \leq m$, $Pr[f_i(S) \neq g_i(S)] = e^{-\Omega(m)}$.*

Proof: Suppose S is a subset of elements and $1 \leq i \leq m$. By our construction, $f_i(S) \leq g_i(S)$. Therefore $Pr[f_i(S) \neq g_i(S)] = Pr[f_i(S) < g_i(S)]$.

First of all, we claim that the above probability is maximized when $S \subseteq G_i$ and $|S| = (m+1)^2/4$. For this, if $S \not\subseteq G_i$, then $f_i(S) = g_i(S) = \infty$ hence $Pr[f_i(S) < g_i(S)] = 0$. Now suppose $S \subseteq G_i$ and $|S| \geq (m+1)^2/4$. Then $g_i(S) = (m+1)^2/4$. Therefore

$$Pr\left[f_i(S) < g_i(S)\right] \ = \ Pr\left[|S \cap \overline{S_i}| + \min\{|S \cap S_i|, (1+\delta)(m+1)/2\} \ < \ (m+1)^2/4\right]$$

This probability can only increase when we remove elements from S. For the case

when $|S| \leq (m+1)^2/4$, we get:

$$\begin{aligned} Pr[\ f_i(S) < g_i(S)\ ] &= Pr\left[\ |S \cap \overline{S_i}| + \min\{|S \cap S_i|,\ (1+\delta)(m+1)/2\ \} \ <\ |S|\ \right] \\ &= Pr[\ \min\{|S \cap S_i|,\ (1+\delta)(m+1)/2\ \} \ <\ |S \cap S_i|\ ] \\ &= Pr[\ |S \cap S_i| \ >\ (1+\delta)(m+1)/2\ ] \end{aligned}$$

Thus, this probability can only increase when more elements are added to S. Hence under the condition $S \subseteq G_i$, $|S| \leq (m+1)^2/4$, the probability is also maximized when $|S| = (m+1)^2/4$.

Now we assume $S \subseteq G_i$ and $|S| = (m+1)^2/4$. In this case, $Pr[f_i(S) < g_i(S)] = Pr[|S \cap S_i| > (1+\delta)(m+1)/2]$, which by a standard Chernoff bound arguments, can be shown to be bounded by $e^{-\Omega(m)}$. □

If we define $\mathbf{f}(S) = (f_1(S), ..., f_m(S))$ and $\mathbf{g}(S) = (g_1(S), ..., g_m(S))$, then by a simple union bound, as a corollary of the lemma, we have $Pr[\mathbf{f}(S) \neq \mathbf{g}(S)] = poly(m)e^{-\Omega(m)}$. Now suppose $\mathcal{A}$ is a deterministic algorithm which makes polynomially many queries to the value oracle. Then by the union bound, with probability at most $poly(m) \cdot e^{-\Omega(m)}$, $\mathcal{A}$ can distinguish between $\mathbf{f}$ and $\mathbf{g}$. Notice that for the cost function $\mathbf{f} = (f_1, \cdots, f_m)$, the optimal solution is at most $(1+\delta)m(m+1)/2$ achieved by assigning S_i to agent i. However, as we showed, the optimal solution for the cost function $\mathbf{g} = (g_1, g_2, \cdots, g_m)$ has cost at least $d(m+1)^2/4$, thus with high probability, $\mathcal{A}$ can not approximate a combinatorial reverse auction instance within factor $\frac{(m+1)^2 d/4}{(1+\delta)m(m+1)/2} \simeq d = c \log n$ for some $c < 1$.

At last, by Yao's principle, we have the following:

Theorem 2.3.2 *A randomized approximation algorithm for the Combinatorial Reverse Auction problem within factor $c \log n$ for some $c < 1$ needs to make exponentially many value queries.*

2.3.2 A $\min(m, \log n)$ approximation algorithm for combinatorial reverse auction

A $\log n$-approximate algorithm for this problem appeared in [25]. In what follows we provide a $\min(m, \log n)$ approximation algorithm. Consider the following LP relaxation (LP1) and its dual (LP2).

$$\begin{array}{llll} \min \sum_{S \subseteq V} \sum_{i} x_{i,S} f_i(S) & \text{(LP1)} & \max \sum_{u \in X} y_u & \text{(LP2)} \\ \sum_{S: u \in S} \sum_{i} x_{i,S} \geq 1 & \forall u \in X & \sum_{u \in S} y_u \leq f_i(S) & \forall S \subseteq V,\ \forall i \\ x_{i,S} \geq 0 & \forall S \subseteq V,\ \forall i & y_u \geq 0 & \forall u \in X \end{array}$$

In LP1, $x_{i,S}$ is used to represent the fraction of set S that is allocated to agent i. Since $f_i(S) - \sum_{u \in S} y_u$ is a submodular function, we can construct a separation oracle for the dual program using the submodular minimization algorithm as a subroutine. Thus we can solve LP1 and LP2 optimally. The following lemma describes the structure of an optimal solution to LP1.

Lemma 2.3.3 *There exists an optimal fractional solution to LP1 such that for every agent i the set $\mathcal{T}_i = \{\ S\ :\ x_{i,S} > 0\ \}$ forms a nested family.*

Proof: Let x be any feasible solution to LP1. If $\mathcal{T}_i$ is not nested, then there exist $A, B \in \mathcal{T}_i$ such that neither A nor B is contained in the other. We may assume $x_{i,A} \geq x_{i,B}$. We will construct another feasible solution x' to LP1 as follows:

- $x'_{i,A \cup B} = x_{i,B}$
- $x'_{i,A} = x_{i,A} - x_{i,B}$
- $x'_{i,B} = 0$
- $x'_{i,A \cap B} = x_{i,B}$ if $A \cap B \neq \emptyset$
- $x'_{i,S} = x_{i,S}$ for all other $S \in X$.
- $x'_{j,S} = x_{j,S}\ \forall j \neq i$ and $\forall S \in X$

By submodularity, one can verify that the cost of the solution x' is at most the cost of x. If the set $\mathcal{T}'_i$ corresponding to x' is nested, we are done. Otherwise, we repeat the procedure for x'. The termination of the above procedure can be guaranteed by

observing that the potential function $\sum_{S \in \mathcal{T}_i} |S|^2$ strictly increases and is polynomially bounded. $\square$

Let $\overline{x}$ be an optimal solution of LP1 which satisfies the conditions in lemma 2.3.3 and $\mathcal{T}_i$ be the corresponding nested families of sets. Let W be the cost of the optimal solution, and also let $\mathcal{T} = \bigcup_i \mathcal{T}_i$. Let Y denote the set of uncovered elements in X. In each iteration pick the set $(i, S) \in \mathcal{T}$ minimizing $f_i(S)/|S \cap Y|$. Add S to the cover and assign it to agent i. Remove all the newly covered elements from Y. Repeat until all elements are covered. Since each $\mathcal{T}_i$ is a nested family, an agent can drop all but the largest set assigned to her. Let (i, S) be the set covering an element u in the integral cover, and let Y be the set of uncovered elements at that point. Then we define $\alpha(u) = f_i(S)/|S \cap Y|$ to be the cost 'borne' by u. Note that $\sum_u \alpha(u)$ is exactly the cost of the integral cover.

Let $u \in X$ be the j'th element to be covered by this algorithm and let (i, S) be the set chosen to cover it. Suppose u was picked during the algorithm. Then since $\overline{x}$ is a fractional cover of Y, $f_i(S)/|S \cap Y| \leq W/|Y|$.

$$\alpha(u) \leq \frac{f_i(S)}{|S \cap Y|} \leq \frac{W}{|Y|} = \frac{W}{(|X| - j + 1)}$$

On the other hand, if u was not picked by the algorithm, then $\alpha(u) \leq W/(|X| - j' + 1) \leq W/(|X| - j + 1)$ for some $j' < j$.

Summing over all u, we conclude that the integral cover has cost at most $W \log n$. To prove that this algorithm is also m-approximate, observe that each set selected has cost at most W. Moreover, each agent is assigned at most one set in the final solution. This proves the claim.

2.4 Vertex Cover

In this section, we consider the submodular vertex cover problem. We first prove an information theoretic lower bound of $2 - \epsilon$ (for any fixed ϵ) for the single agent case

and provide an algorithm with approximation ratio of 2. We then present a $2 \log n$ approximation algorithm for the multi-agent case and an information theoretic lower bound of $\Omega(\log n)$.

2.4.1 Single agent case

We are given an undirected graph $G(V, E)$ and a normalized monotone submodular function $f : 2^V \longrightarrow \mathbb{R}$. We wish to find a vertex cover $U \subseteq V$ of graph G such that $f(U)$ is minimized.

Theorem 2.4.1 *For every fixed $\epsilon > 0$, any randomized algorithm for the submodular vertex cover problem with an approximation ratio of $2 - \epsilon$ needs exponentially many queries to the value oracle.*

Proof: Consider a bipartite graph $G(A \cup B, E)$ such that $|A| = |B| = n$. The edge set consists of n edges which forms a matching between A and B. Let R be a random minimum cardinality vertex cover of this graph, which can be picked by choosing one endpoint of every edge uniformly at random.

Define the following two submodular cost functions.

$$\begin{aligned} f_R(S) &= \min \left\{ \; |S \cap \overline{R}| \; + \min \left\{ \; |S \cap R|, \; \frac{(1+\delta)n}{2} \; \right\}, \; n \; \right\} \\ g(S) &= \min \{ \; |S|, \; n \; \} \end{aligned}$$

Here δ is chosen such that $2/(1+\delta) = 2 - \epsilon$. Notice that the optimum value of the vertex cover for the function f_R is $\frac{(1+\delta)n}{2}$, and for g it is n. Thus if we can show that any randomized algorithm, cannot distinguish between f_R and g with high probability, it will imply an inapproximability ratio of $2/(1+\delta)$ or $2 - \epsilon$ for the submodular vertex cover problem.

As discussed earlier, it suffices to show that for a deterministic query Q, $Pr[\, f_R(Q) \neq g(Q) \,]$ is exponentially small, where the probability space is defined over the random choice of set R. Since $f_R(S) \leq g(S)$ for all $S \subseteq V$, $f_R(Q) \neq g(Q)$ implies

$f_R(Q) < g(Q)$.

Let Q^* be the optimal query for which $Pr[\ f_R(Q) < g(Q)\ ]$ is maximized. We will show that $|Q^*| = n$. First, suppose that $|Q| \geq n$, then

$$\begin{aligned} Pr[\ f_R(Q) < g(Q)\ ] &= Pr[\ f_R(Q) \ < \ n\ ] \\ &= Pr\left[\ |Q \cap \overline{R}| \ + \ \min\{\ |Q \cap R|,\ (1+\delta)n/2\ \} \ < \ n\ \right] \end{aligned}$$

which increases as the size of Q is reduced. Thus the size of the optimal query in this case is n.

Now suppose $|Q| \leq n$. In this case,

$$\begin{aligned} Pr[\ f_R(Q) < g(Q)\ ] &= Pr[\ f_R(Q) < |Q|] \\ &= Pr\left[\ |Q \cap \overline{R}| \ + \ \min\{\ |Q \cap R|,\ (1+\delta)n/2\ \} \ < \ |Q|\ \right] \\ &= Pr\left[\ \min\{\ |Q \cap R|,\ (1+\delta)n/2\ \} \ < \ |Q \cap R|\ \right] \\ &= Pr[\ |Q \cap R| \ > \ (1+\delta)n/2\ ] \end{aligned}$$

which increases as $|Q|$ is raised. Therefore, the optimal query size in this case is also n.

Hence $|Q^*| = n$. Let k be the number of edges for which both the end points are contained in Q^* and Q_1 be the set of these endpoints ($|Q_1| = 2k$). Let $Q_2 = Q^* - Q_1$. We have:

$$\begin{aligned} Pr[\ f_R(Q^*) < g(Q^*)] &= Pr\left[\ |Q^* \cap R| \ > \ (1+\delta)\frac{n}{2}\ \right] \\ &= Pr\left[\ |Q_2 \cap R| \ > \ (1+\delta)\frac{n}{2} - k\ \right] \\ &= Pr\left[\ |Q_2 \cap R| \ > \ (1+\delta)\frac{|Q_2|}{2} + \delta k\ \right] \qquad (1) \end{aligned}$$

If $\delta k \geq (1-\delta)\frac{|Q_2|}{2}$, then the expression in equation (1) reduces to $Pr[|Q_2 \cap R| > |Q_2|] \ = \ 0$. On the other hand if $\delta k < (1-\delta)\frac{|Q_2|}{2}$, then

$$|Q_2| \ = \ n - 2k \ > \ n - \frac{1-\delta}{\delta}|Q_2|$$

which implies $|Q_2| > \delta n$. Every vertex in Q_2 belongs to R with probability $\frac{1}{2}$ with independence, and $E[|Q_2 \cap R|] = |Q_2|/2 = \delta n/2$. Therefore, applying Chernoff bounds:

$$\begin{aligned} Pr[\ f_R(Q^*) < g(Q^*)] &= Pr\left[\ |Q_2 \cap R| \ > \ (1+\delta)\frac{|Q_2|}{2} + \delta k\ \right] \\ &\leq Pr\left[\ |Q_2 \cap R| \ > \ (1+\delta)\frac{|Q_2|}{2}\right] \\ &\leq e^{-\frac{\delta^3 n}{2}} \end{aligned}$$

Hence, the probability that an arbitrary query Q can distinguish between f and g is exponentially small. □

Theorem 2.4.2 *There exists an algorithm which finds a 2-approximate solution to the single agent vertex cover problem with submodular costs.*

Proof: We formulate the problem as a configurational LP and round the fractional solution. Let variable x_S be an indicator variable for the set S of vertices being the vertex cover. Then the following LP is a lower bound on the value of the optimal integral solution.

$$\begin{array}{rlrl} \min \sum_{S \subseteq V} x_S f(S) & (LP3) & \max \sum_{e \in E} y_e & (LP4) \\ \sum_{S:u \in S} x_S + \sum_{S:v \in S} x_S \geq 1 & \forall (u,v) \in E & \sum_{v \in S} \sum_{e \in \delta(v)} y_e \leq f(S) & \forall S \subseteq V \\ x_S \geq 0 & \forall S \subseteq V & y_e \geq 0 & \forall e \in E \end{array}$$

It is not difficult to see that the function $\sum_{v \in S} \sum_{e \in \delta(v)} y_e$ is a modular function. Thus $f(S) - \sum_{v \in S} \sum_{e \in \delta(v)} y_e$ is a submodular function, and we can use the submodular minimization algorithm as a subroutine to construct a separation oracle for the dual. This allows us to find an optimal fractional solution to LP3 with value at most OPT. Let x^* be this solution. Output $Q = \left\{\ u \in V \ : \ \sum_{S:u \in S} x^*_S \geq 1/2\ \right\}$ as the vertex

cover. Clearly, for any $(u, v) \in E$, either $\sum_{S:u\in S} x^*_S \geq 1/2$ or $\sum_{S:v\in S} x^*_S \geq 1/2$ must hold, thus Q is a valid vertex cover of G. Now, for monotone submodular functions it is known that if a fractional solution y_S $(\forall S \subseteq V)$ covers a set Q, in the sense that $\sum_{S:i\in S} y_S \geq 1$ for all $i \in Q$, then $f(Q) \leq \sum_{S\subseteq V} y_S * f(S)$. Thus, since $2x^*$ is a fractional cover of Q, we get that $f(Q) \leq 2\sum_{S\subseteq V} x^*_S * f(S) = 2 \cdot \text{OPT}$. □

2.4.2 Multi-agent Case

We are given an undirected graph $G(V, E)$ and a normalized monotone submodular function $f_i : 2^V \longrightarrow \mathbb{R}$ for each agent i. We wish to find a vertex cover $U \subseteq V$, and a partition $U_1, U_2, \cdots, U_k$ of U such that $\sum_i f_i(U_i)$ is minimized.

A lower bound of $\Omega(logn)$ for the multi-agent case follows easily from the lower bound of the reverse auction setting. This is because even if we know the set of vertices which form the vertex cover, it is hard (in information-theoretic sense) to find optimal allocation of these vertices to the agents - meaning which agent builds which set of vertices of the given vertex cover. Thus we get the following theorem:

Theorem 2.4.3 *Any randomized algorithm for the multi-agent submodular vertex cover problem with an approximation ratio $c \log n$ for some constant $c < 1$ needs exponentially many queries to the value oracle.*

2 log n-approximate algorithm: We begin by finding an optimal fraction solution $\overline{x}$ using the LP relaxation LP5, which gives a lower bound on the optimal integral solution. The given LP can be solved by constructing a separation oracle of the dual program as shown earlier in the single agent case. Consider the set $Q = \left\{ u \in V \ : \ \sum_{S:u\in S} \sum_i \overline{x}_{i,S} \geq 1/2 \right\}$ which forms a valid vertex cover. We will now round $2\overline{x}$ to find an allocation of vertices in Q to the various agents. Let W denote the total cost of the solution $2\overline{x}$.

$$\min \sum_{S \subseteq V} \sum_i x_{i,S} f_i(S) \qquad (LP5)$$

$$\sum_{S:u \in S} \sum_i x_{i,S} + \sum_{S:v \in S} \sum_i x_{i,S} \geq 1 \qquad \forall (u,v) \in E$$

$$x_{i,S} \geq 0 \qquad \forall S \subseteq V,\ \forall i$$

The algorithm proceeds in rounds; in each round we cover some elements of the set Q. Let at the beginning of round t of the algorithm, Z_t be the set of uncovered elements in Q. For any fractional cover x of set Z and $u \in Z$, define $\alpha_{x,Z}(u) = \sum_{S:u \in S} \sum_i \frac{x_{i,S} f_i(S)}{|S \cap Z|}$. Note that $\sum_{u \in Z} \alpha_{x,Z}(u) = \sum_{i,S:S \cap Z \neq \phi} x_{i,s} f_i(S) \leq \sum_{i,S} x_{i,s} f_i(S)$. Now let u_t be an element in Z_t that minimizes $\alpha_{2\overline{x},Z_t}(u)$. Among the sets containing u_t, choose a set (i,S) randomly with probability proportional to $2 * \overline{x}_{i,S}$. Remove all the newly covered elements from Z_t and call the new subset Z_{t+1} and iterate until all the elements are covered. Set $y_{i,S} = 1$ if (i,S) was picked in any round of the algorithm.

Analysis: Let $u_1, u_2, ...$ be the order in which the vertices of Q get covered. We claim that $E[\alpha_y(u_j)] \leq W/(|Q| - j + 1)$. Suppose u_j was picked during the algorithm. Then, $E[\alpha_y(u_j)] \leq \alpha_{2\overline{x}}(u_j)$. Since $2\overline{x}$ covers the remaining $|Q| - j + 1$ elements in Q, $\alpha_{2\overline{x}}(u_j) \leq W/(|Q| - j + 1)$. On the other hand, if u_j was not picked during the algorithm, then

$$E[\alpha_y(u_j)] \ = \ \alpha_{2\overline{x}}(u'_j) \ \leq \ \frac{W}{(|Q| - j' + 1)} \ \leq \ \frac{W}{(|Q| - j + 1)}$$

for some $j' < j$. Summing over j, we have

$$\sum_{i,S} y_{i,S} f_i(S) \ = \ \sum_{u \in Q} \alpha_y(u) \ \leq \ \sum_{u \in Q} \alpha_{2\overline{x}}(u) \ \leq \ W \log n \ \leq \ 2\mathrm{OPT} \cdot \log n$$

This algorithm can be derandomized using standard techniques.

2.5 Shortest Path

In this problem we are given an undirected graph $G(V, E)$ and a monotone submodular cost function $f_i : 2^E :\longrightarrow \mathbb{R}$ for each agent i. The goal is to find a path P between two given vertices, and partition of P into $P_1, P_2, \cdots, P_k$ such that $\sum_i f_i(P_i)$ is minimized. We first consider the single agent case and provide an information-theoretic lower bound of $\Omega(n^{2/3})$ for all fixed $\epsilon > 0$, ignoring poly-logarithmic factors. We also present an $O(n^{2/3})$-approximation algorithm for this problem. Lastly, we comment on the gap that exists between the upper and lower bounds for the multi-agent case, in context of our results for the single agent case.

2.5.1 Single agent case

As in previous sections, we proceed by designing two submodular functions that are hard to distinguish in polynomially many queries but have different optimal values. In the general framework outlined in section 2.2, this is accomplished by 'hiding' a random element of lower cost from the target collection C in one of the functions. In this case, C is the set of all $s-t$ paths. However an identical analysis does not work in this case. This is because for a pair of adjacent edges, the events that these edges belong to the random shortest $s-t$ path are are not independent precluding the use of Chernoff bounds which makes the analysis a lot more involved. In this section we use a simple pigeon hole principle argument to solve this problem.

Theorem 2.5.1 *Any randomized approximation algorithm for the submodular shortest path problem with factor $O\left(\frac{n^{2/3}}{\log n}\right)$ needs super-polynomially many queries.*

Proof: Consider the graph G which is a level graph having $n^{2/3}+2$ levels of vertices. First level contains only vertex s and the last level contains only t. Each other level has $n^{1/3}$ vertices and there exists a complete bipartite graph between successive levels. Let R be a randomly chosen $s-t$ path of length $n^{2/3}+1$.

Define the following two submodular cost functions $f, g : 2^E \longrightarrow \mathbb{R}^+$:

$$\begin{aligned} f(Q) &= \min\{\ |Q \cap \overline{R}|\ +\ \min\{\ |Q \cap R|,\ \log n\ \},\ n^{2/3}+1\ \} \\ g(Q) &= \min\{\ |Q|,\ n^{2/3}+1\ \} \end{aligned}$$

Clearly, the ratio of optima in g and f is $\Omega\left(\frac{n^{2/3}}{\log n}\right)$.

To prove the lower bound it suffices to prove that $Pr[f(Q) \neq g(Q)]$ is super-polynomially small for an arbitrary query Q. This happens if and only if $f(Q) < g(Q)$. Making arguments analogous to the proof of theorem 2.4.1, $Pr[\ f(Q) < g(Q)\ ]$ is maximized when $|Q| = 1 + n^{2/3}$. Therefore,

$$Pr[\ f(Q) < g(Q)\ ] = Pr[\ |Q \cap R| > \log^2 n\ ]$$

Let E_{even} and E_{odd} be the set of edges which are at distance even and odd respectively from the vertex s. Define $Q_{even} = Q \cap E_{even}$ and $Q_{odd} = Q \cap E_{odd}$. Similarly define R_{even} and R_{odd}. Without loss of generality, let $|Q_{odd}| \geq |Q_{even}|$. Thus,

$$\begin{aligned} Pr[\ |Q \cap R| > \log^2 n\ ] &= Pr[\ |Q_{even} \cap R_{even}| + |Q_{odd} \cap R_{odd}| > \log n\ ] \\ &\leq 2 \cdot Pr[\ |Q_{odd} \cap R_{odd}| > \frac{\log n}{2}\ ] \end{aligned}$$

Note that the edges in R_{odd} were chosen independently at random since R was chosen uniformly at random. Also $E\left[|Q_{odd} \cap R_{odd}|\right] = O(1)$. Thus by chernoff bounds we conclude that is $Pr\left[\ |Q \cap R| > \log n\ \right] \leq O\left(e^{-\Omega(\log^2 n)}\right)$, which is super-polynomially small. This proves the theorem. □

Theorem 2.5.2 *There exists an algorithm which finds an $O(n^{2/3})$ approximate solution to the single agent shortest path problem with submodular costs.*

Proof: We begin with two simple approaches to get an $O(n)$-approximate algorithm. Interestingly, we can combine the two ideas to obtain an $O(n^{2/3})$-approximate algorithm for the problem.

Goemans *et al* [21] address the problem of finding approximate explicit representations for submodular functions. They use an ellipsoidal approximation of the polymatroid of the submodular function $f : 2^X \rightarrow \mathbf{R}_+$ to assign a weight w_e to every element $e \in X$. The approximated cost of a set is then defined as $\hat{f}(S) = \sqrt{\sum_{e \in S} w_e}$. They prove that for all S,

$$\hat{f}(S) \;\leq\; f(S) \;\leq\; \sqrt{|X|}\hat{f}(S)$$

and therefore, $\hat{f}$ can be though of as an explicit approximate representation of f. To solve our submodular shortest path problem, a possible approach would be to find the weights w_e for all the edges of the graph and then find the path P minimizing $\hat{f}(P)$. This can be done in polynomial time, since minimizing $\hat{f}(P)$ is the same as minimizing $\left(\hat{f}(P)\right)^2$, which just reduces to the shortest path problem with linear costs. This approach yields a $O(\sqrt{E})$ approximation algorithm. For dense graphs this factor can be as bad as $\Omega(n)$. This method can be useful if the given graph has few edges.

Another simple algorithm to get an $O(n)$ approximation for this problem is to 'guess' the cost of the heaviest edge e in the path, use that as a lower bound on OPT. Define *cost* of an edge e as $f(\{e\})$. The algorithm runs in multiple phases. In each phase choose a new edge and drop all edges that weigh more than the given edge and return any $s - t$ path(if it exists) in the pruned graph. We finally select the smallest $s - t$ paths among those returned during the phases. It is easy to see that this is an $O(l)$ approximate algorithm where l is the number of edges in the optimal path. Once again this can be as bad as $O(n)$ for some graphs. This approach can be useful if the given graph is *dense*, since sufficiently dense graphs are known to have small diameter.

The central idea of our algorithm is to decompose the graph in to sparse and dense clusters. Then we use the first approach to account for sparse regions of the graph and deal with the dense regions using ideas from the second approach.

The algorithm runs in multiple phases, where after each phase we output a path. Final solution is the minimum cost path among these paths. Each phase is identified by a unique edge in the edge set, thus there are $|E|$ phases. Following are the steps, in order, which constitutes a single phase.

Pruning Step: Let e be the edge corresponding to the current phase. Delete all edges that weigh more than e. Let G_e denote the pruned graph. The phase terminates prematurely if the s and the t are disconnected in G_e.

Separation Step: In this step we partition the edge set into those that are in dense regions of the graph and those which belong to sparse regions. Successively remove vertices from G_e whose degree in the remaining graph is at most $n^{1/3}$. Also remove the edges incident on these vertices and add them to the set S_e. Continue removing vertices until all the remaining vertices have degree more than $n^{1/3}$. Let R_e be the remaining edges in G_e. Edges in S_e belong to the sparse part of the graph while those in R_e constitute the dense part of the graph.

Search Step: Using the algorithm in [21], we find an explicit representation for the function f restricted to the S_e. Redefine the costs for edges in S_e to be the weights returned by the ellipsoidal approximation subroutine and set the cost of each edge in R_e to be a zero. Treating these edge costs as additive quantities, find the shortest $s-t$ path passing through e. Let P_e be this path.

Compression Step: The path returned by the search step might contain too many edges, which could be bad for the algorithm. In this step, we compress the path P_e by replacing some of its subpaths by smaller paths(in terms of number of edges). For this we analyze the intersection of P_e with every connected component of $G\left(V, R_e\right)$. Let H be an arbitrary connected component of $G\left(V, R_e\right)$ and let a be the first vertex where P_e enters H and b be the final vertex that it passes through before leaving H for the last time. Replace the sub-path of P_e between a and b with the shortest path in H(in terms of the number of edges) connecting them(refer to the figure below). Do

this for every connected component of $G(V, R_e)$. Report this modified path as the solution for this phase.

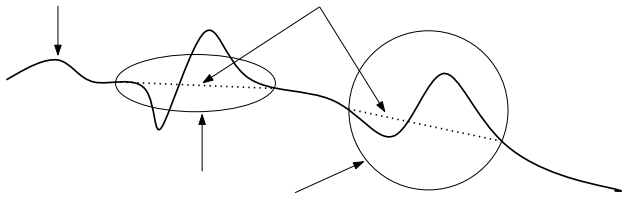

Figure 1: Shortening the path in dense regions

Analysis: To prove that the above algorithm achieves an approximation ratio of $O(n^{2/3})$ we will use the following observation.

Observation 1 *Since all the vertices remaining after the first step have degree greater than $n^{1/3}$, any connected component C of $G(V, R_e)$ has diameter at most $\frac{|V(C)|}{n^{1/3}}$.*

Let P_{OPT} be an optimal path for the problem under the submodular cost function f. Let α be the heaviest edge in this path. Consider the phase corresponding to α. During the separation step of every phase we remove at most $n^{4/3}$ edges since each of the chosen vertices have a degree less than $n^{1/3}$. Thus the subroutine gives an explicit cost function that is a $O(n^{2/3})$ approximation for all subsets of S_α. Let $\hat{f}(A)$ denote the cost of any $A \subseteq S_\alpha$ returned by the subroutine. Thus for all $A \subseteq S_\alpha$, we have:

$$\hat{f}(A) \leq f(A) \leq n^{2/3}\hat{f}(A) \tag{2}$$

Let P_α be the solution returned by this phase. Define $X_\alpha = P_{OPT} \cap S_\alpha$, $Y_\alpha = P_\alpha \cap S_\alpha$ and $Z_\alpha = P_\alpha \cap R_\alpha$. It follows that,

$$\begin{aligned}
2n^{2/3} f(P_{OPT}) &\geq n^{2/3} f(P_{OPT}) + n^{2/3} f(\{\alpha\}) && (3)\\
&\geq n^{2/3} f(X_\alpha) + n^{2/3} f(\{\alpha\}) && (4)\\
&\geq n^{2/3} \hat{f}(X_\alpha) + n^{2/3} f(\{\alpha\}) && (5)\\
&\geq n^{2/3} \hat{f}(Y_\alpha) + n^{2/3} f(\{\alpha\}) && (6)\\
&\geq n^{2/3} \hat{f}(Y_\alpha) + f(Z_\alpha) && (7)\\
&\geq f(Y_\alpha) + f(Z_\alpha) && (8)\\
&\geq f(Y_\alpha \cup Z_\alpha) && (9)\\
&= f(P_\alpha) && (10)
\end{aligned}$$

Equations (3) and (4) follow from monotonicity. Equation (5) uses equation (2). Equation (6) then follows from equation (5) since P_α is the shortest path under the function $\hat{f}$. Also, using the observation above and summing over all components of $G(V, R_\alpha)$ we conclude that $|Z_\alpha|$ can not be more than $n^{2/3}$. Thus equation (7) follows from equation (6) since each edge in $|Z_\alpha|$ costs at most $f(\{\alpha\})$ and f is submodular. We arrive at equations (8), (9), (10) using equation (2) and the submodularity of function f. □

2.5.2 Multi-agent case

One glance at Table 1 reveals the fact that the multi-agent submodular shortest path problem is the only problem left 'open' in the sense that we do not have an algorithm that matches the $\Omega(n^{2/3})$ information theoretic lower bound. We will first explain why this gap exists, and then elaborate on some of the interesting issues it emphasizes.

Obviously, the $\Omega(n^{2/3})$ lower bound from Theorem 2.5.1 also applies in presence of multiple agents. However, our algorithm from the previous section cannot be ported to the multi-agent case. Recall the two basic approaches we outlined that yield $O(n)$

approximations to the single agent submodular shortest path problem: 1) Pruning the graph of edges heavier than the heaviest edge in the optimal solution and 2) Ellipsoidal approximation of the submodular function as provided in [21]. While we can still use the former to obtain an $O(n)$-approximation in the multi-agent case, the latter fails due to inherent computational hardness. Finding a path P that minimizes the ellipsoidal approximation function $\hat{f}(P)$ was computationally feasible, because minimizing $\hat{f}(P)$ is equivalent to minimizing $\left(\hat{f}(P)\right)^2$, which reduces to finding the shortest path under linear costs. For this approach to work in the multi-agent case however, we need to find an $s-t$ path P, and partition it into the m agents as $P_1, ..., P_m$ such that $\sum_i \hat{f}_i(P_i)$ is minimized. This function is not linear in terms of the ellipsoidal weights, and in particular is known to be NP-hard to minimize.

It is important to note however, that the lower bound established by Theorem 2.5.1 is the best possible information theoretic hardness result. Recall that such lower bounds only limit the number of calls made to the value oracle, and no restriction is placed on the computational complexity of the algorithm outside of the oracle calls. Indeed, it is easy to generalize the algorithm in the previous section to the multi-agent case, if we have the computational capacity to minimize the non-linear objective function discussed above. Therefore, there does exist an algorithm that makes polynomially many calls to the value oracle, performs exponentially many other computational operations and guarantees an approximation factor of $O(n^{2/3})$. This indicates that a stronger hardness result needs to combine the computational and information theoretic complexity of the problem into one argument. In the next chapter, we will study such hardness results for the class of discounted price functions.

2.6 Perfect Matching

In this section, we consider the multi-agent submodular minimum perfect matching problem. In this problem we are given a bipartite graph $G(V, E)$ where $|V| = n$, containing at least one perfect matching and a normalized monotone submodular function $f_i : 2^E \longrightarrow R^+$ for each agent i. We wish to find a perfect matching M, and a partition of M into $M_1, M_2, \cdots, M_m$ such that $\sum_i f_i(M_i)$ is minimized. We first prove an information theoretic lower bound of $\Omega(n)$ on the approximability of the single agent case, which also implies the same bound for the multi-agent case. Then, we give an n-approximate algorithm for the multi-agent case.

As in previous sections, we proceed by designing two submodular functions that are hard to distinguish in polynomially many queries but have widely differing optimal values. In the general framework outlined in section 2.2, this is accomplished by 'hiding' a random element of lower cost from the target collection C in one of the functions. In this case, C is the set of all perfect matchings. Once again choosing a random matching from C however does not serve our purpose because for a fixed pair of edges, the events that these edges belong to the random matching are not independent, thus precluding the use of Chernoff bounds. We circumvent this problem by using the following result from the theory of random graphs [7]:

Lemma 2.6.1 *Let $G(n, n, p)$ be a random bipartite graph on $2n$ vertices such that each edge is present independently with probability p. Then*

$$Pr[\ G(n, n, p)\ \textit{contains no perfect matching}\ ] = O(ne^{-np})$$

Now instead of hiding a randomly chosen perfect matching, we hide a collection of randomly and independently chosen edges that contains a perfect matching with high probability. We prove the following theorem.

Theorem 2.6.2 *Any randomized approximation algorithm for the submodular minimum cost perfect matching problem with factor $O\left(\frac{n}{\log^2 n}\right)$ needs super-polynomially*

many queries. There exists an algorithm that approximately finds an n-approximate minimum cost matching in polynomial time.

Proof: Consider the complete bipartite graph $K_{n,n}$. We choose a random subset R of edges by picking each edge independently with probability $p = \log^2 n/n$. By applying lemma 2.6.1, the probability that R does not contain a perfect matching is $O\left(ne^{-\log^2 n}\right)$, which is super-polynomially small.

Define the following two submodular cost functions $f_R, g : 2^E \longrightarrow \mathbb{R}^+$:

$$\begin{aligned} f_R(Q) &= \min\{\ |Q \cap \overline{R}|\ +\ \min\{\ |Q \cap R|,\ (1+\delta)\log^2 n\ \},\ n\ \} \\ g(Q) &= \min\{\ |Q|,\ n\ \} \end{aligned}$$

With probability $1 - O(ne^{-\log^2 n})$, R contains a perfect matching and hence the minimum cost of a perfect matching in f is at most $(1+\delta)\log^2 n$. Therefore the ratio of optima in g and f is $\Omega\left(\frac{n}{\log^2 n}\right)$ with high probability.

Now we look at the probability that the algorithm can not distinguish f and g. It suffices to prove that $Pr[f_R(Q) \neq g(Q)]$ is super-polynomially small for an arbitrary query Q. It's easy to see that $f_R(S) \leq g(S)$, thus these two functions differ on Q if and only if $f_R(Q) < g(Q)$.

Making arguments analogous to the proof of theorem 2.4.1, $Pr[\ f_R(Q) < g(Q)\ ]$ is maximized when $|Q| = n$. Therefore,

$$Pr[\ f_R(Q) < g(Q)\ ] = Pr[\ |Q \cap R| > (1+\delta)\log^2 n\ ]$$

Since $E[|Q \cap R|] = \log^2 n$ and edges were picked uniformly at random, we can apply Chernoff bounds to conclude that this probability is $O(e^{-\delta^2 \log^2 n})$. This proves the theorem. □

Factor n approximation algorithm: We are given a graph $G(V, E)$ and submodular cost functions f_i for each agent. Define a new cost function w over E as

$w_e = \min_i f_i(\{e\})$ and define $w(Z) = \sum_{e \in Z} w_e$ for all $Z \subseteq E$. Since w is an additive valuation function we can find a minimum value perfect matching in polynomial time. Let M be such a matching. Assign each edge $e \in M$ to the agent having the minimum valuation for that edge. Let the cost of this solution under the original valuation functions be W.

We now prove that this is an n-approximate algorithm. By submodularity we have $W \leq w(M)$. Let M_0 be an optimal solution having value OPT. Since M is a minimum weight matching under w, $w(M) \leq w(M_0)$.

Let $w_{max} = \max_{e \in M_0}\{f_i(e) \mid \text{e assigned to agent } i \text{ } in \text{ } M_0\}$. By submodularity of the cost functions, $w(M_0) \leq n \cdot w_{max}$. By monotonicity we have $w_{max} \leq OPT$. Therefore,

$$W \leq w(M) \leq w(M_0) \leq n \cdot w_{max} \leq n \cdot OPT$$

This completes the analysis.

2.7 Spanning Tree

In this section, we consider the multi-agent submodular minimum spanning tree problem. We are given a connected graph $G(V, E)$ where $|V| = n$ and a normalized monotone submodular function $f_i : 2^E \longrightarrow R^+$ for each agent i. We want to find a spanning tree T of G, and a partition of T into $T_1, T_2, \cdots, T_m$ such that $\sum_i f_i(T_i)$ is minimized. We first prove an information theoretic lower bound of $\Omega(n)$ on the approximability of the single agent case, which also implies the same bound for the multi-agent case. Then, we give an n-approximate algorithm for the multi-agent case.

To prove the lower bound we will provide two submodular functions that can not be distinguished in polynomially many queries and have widely differing optimal values. As in Section 2.6, we will use the following lemma [7] in the proof.

Lemma 2.7.1 *Let $G(n, p)$ be a random graph on n vertices such that each edge is*

present independently with probability p. Then

$$Pr[\ G(n,p)\ \textit{is disconnected}\ ] \leq n(1-p)^n.$$

Theorem 2.7.2 *Any randomized approximation algorithm for the submodular minimum spanning problem on a with factor $O\left(\frac{n}{\log^2 n}\right)$ needs super-polynomially many queries. There exists an algorithm that approximately finds an n-approximate spanning tree in polynomial time.*

Proof: Consider K_n, the clique graph on n vertices. We choose a random subset of edges R, by picking each edge independently with probability $p = \log^2 n/n$. By applying Lemma 2.7.1, the probability that R is not connected is $O\left(ne^{-\log^2 n}\right)$, which is super-polynomially small.

Define the following two submodular cost functions $f_R, g : 2^E \longrightarrow \mathbb{R}^+$:

$$\begin{aligned} f_R(Q) &= \min\{\ |Q \cap \overline{R}|\ +\ \min\{\ |Q \cap R|,\ (1+\delta)\log^2 n\ \}\ ,\ n\ \} \\ g(Q) &= \min\{\ |Q|,\ n\ \} \end{aligned}$$

With probability $1 - O(ne^{-n^\epsilon})$, R is connected and hence, the cost of the optimal spanning tree in f is at most $(1+\delta)\log^2 n$. Therefore, the ratio of optimal solution values in g and f is $\Omega\left(\frac{n}{\log^2 n}\right)$ with high probability.

Making arguments similar to the proof of Theorem 2.6.2, we conclude that $Pr[\ f_R(Q) < g(Q)\ ] = O(ne^{-\delta^2 \log^2 n})$ for any query Q. This suffices to prove the theorem.

Factor n approximation algorithm: We are given a graph $G(V,E)$ and submodular cost functions f_i for each agent. Define a new cost function w over E as $w_e = \min_i f_i(\{e\})$. Run Kruskal's algorithm on G treating w_e as the cost of the edge e to get a minimum spanning tree T. Assign each edge $e \in T$ to the agent i minimizing $f_i(\{e\})$. The proof that this constitutes an n-approximate solution follows similar arguments as the analysis of the n-approximate algorithm for perfect matching. □

2.8 Summary

In this chapter, we study several fundamental submodular combinatorial optimization problems in the oracle model. We obtain information-theoretic lower bounds on their approximabilities and design matching algorithm for each problem. We will study the computational aspect of submodular combinatorial optimization under the discounted price model in the next chapter .

CHAPTER III

SUBMODULAR COMBINATORIAL OPTIMIZATION: DISCOUNTED PRICE MODEL

3.1 Introduction

In the previous chapter, we studied the submodular combinatorial optimization problems in the oracle model. Unfortunately, many of the fundamental optimization problems have turned out to be extremely hard if we are given only value oracles. Thus, from a practical standpoint, the applicability of these results is not very well-founded as the class of submodular functions might be much more general than real-world functions. Moreover, the class of submodular functions is defined over an exponentially large domain and thus requires exponential time to write down the function explicitly. This may not be the case in real-world applications.

In this chapter, we wish to explore functions that lie between the additive functions and the general submodular functions, and that are also succinctly representable. In particular, we study *discounted price functions* in which we are given an additive function c and a discount function $d : R^+ \to R^+$ that is a concave curve. The price of any subset S is defined to be $d(c(S))$. It is not difficult to see that discounted price functions form a subclass of submodular functions.

The practical motivation of discounted price function is that agents usually give discounts to their prices for a subset of items. On the other hand, these functions have strong theoretical motivation as well. A common technique (due to [21]) for designing optimal algorithms under general submodular functions is to first approximate submodular functions by *ellipsoid* functions. These ellipsoid functions form a special class of discounted price functions.

3.1.1 Discounted Price Model

We define a function $d :\rightarrow \mathbb{R}^+$ to be a discounted price function if it satisfies the following properties: (1) $d(0) = 0$; (2) d is increasing; (3) $d(x) \leq x$ for all x; (4) d is concave.

We study combinatorial problems in the following general setting. We are given a set of elements E, and a collection Ω of its subsets. We are also given a set $\mathcal{A}$ of k agents where each agent $a \in \mathcal{A}$ specifies a cost function $c_a : E \rightarrow \mathbb{R}^+$ where $c_a(e)$ indicates her cost for the element e. Each agent also declares a discounted price function d_a. If an agent a is assigned a set T of elements, then her total price is specified by $d_a(\sum_{e \in T} c_a(e))$. This is called her *discounted price*. For the ease of notation we will use $d_a(T)$ to denote $d_a(\sum_{e \in T} c_a(e))$. The objective is to select a subset S from Ω and a partition $S_1, S_2, ..., S_k$ of S, such that $\sum_{a \in \mathcal{A}} d_a(S_a)$ is minimized.

We study the following four problems over an undirected graph $G(V, E)$.

- **Discounted Edge Cover**: In this problem, Ω is chosen to be the collection of edge covers.
- **Discounted Spanning Tree**: In this problem, Ω is the collection of spanning trees of the graph.
- **Discounted Perfect Matching**: In this problem, we assume the graph has an even number of vertices. Ω is chosen to be the collection of perfect matchings of the graph.
- **Discounted $s-t$ Path**: In this problem, we are given two fixed vertices s and t. Ω consists of all paths connecting s and t.

3.1.2 Our Results, Organization and Related Work

In section 3.2.1, we show that the discounted edge cover and spanning tree problems are hard to approximate within a factor of $(1 - o(1)) \log n$ unless P = NP. In section

3.2.2 and 3.2.3, we amplify this result to show that $s-t$ path and matching are hard to approximate within any polylog factor.

On the algorithmic front, in section 3.3.1 and 3.3.2, we show that our results are tight by giving $\log n$-approximate algorithms for the discounted edge cover and spanning tree problems. In section 3.3.3, we describe simple $O(n)$-approximate algorithms for discounted $s-t$ and perfect matching. We leave the design of sublinear approximation algorithms for these two problems as an open question.

The results in this chapter is a follow up work of submodular combinatorial optimization in the oracle model studied in chapter 2. Independent of our work [18], recently, Iwata and Nagano [29] also also study submodular cost set cover and submodular edge cover problems. At last, much of the other related work has been listed in section 2.1.2 from chapter 2 and we omit for here.

3.2 Hardness of Approximation

In this section we present hardness of approximation results for the four problems defined earlier. Unlike some of the previous work on combinatorial optimization [18, 49] over non-linear cost functions, the bounds presented here are not information theoretic but are contingent on P $\neq$ NP. In section 3.2.1, we show that all our problems are hard to approximate within factor $\log n$. In section 3.2.2 and 3.2.3, we amplify the hardness of approximation for discounted $s-t$ path and perfect matching to $O(\log^c n)$ for any constant c.

Recall that in each of the problems we are given a graph $G = (V, E)$ over n vertices. We are also given a set $\mathcal{A}$ of k agents each of whom specifies a cost $c_a : E \to \mathbb{R}^+$. Here $c_a(e)$ is the cost for building edge e for agent a. Each agent also specifies a discounted price function given by $d_a : \mathbb{R}^+ \to \mathbb{R}^+$. The objective is to build a specified combinatorial structure using the edges in E, and allocate these edges among the agents such that the sum of discounted prices for the agents is minimized.

3.2.1 Basic Reduction

To show the logarithmic hardness of approximation for the problems stated earlier we consider the following general problem and use a reduction from set cover to establish its hardness.

Discounted Reverse Auction: We are given a a set E of n items and a set $\mathcal{A}$ of agents each of whom specifies a function $c_a : E \to \mathbb{R}^+$. Here $c_a(e)$ is the cost for procuring item e from agent a. Each agent also specifies a discounted price function given by $d_a : \mathbb{R}^+ \to \mathbb{R}^+$. The task is to find a partition $\mathcal{P} = \{P_1 \cdots P_k\}$ of E such that $\sum_{a \in \mathcal{A}} d_a(\sum_{e \in P_a} c_a(e))$ is minimized.

Lemma 3.2.1 *It is hard to approximate the discounted reverse auction problem within factor* $(1 - o(1)) \log n$ *unless* P = NP.

Proof: We reduce set cover to the discounted reverse auction problem to prove this result. Consider an instance $\mathcal{I} = (U, C, w)$ of set cover where we wish to cover all elements in the universe U using sets from C and minimize the sum of weights under the weight function $w : C \to \mathbb{R}^+$. We define an instance, $\mathcal{I}'$ of our discounted reverse auction problem corresponding to $\mathcal{I}$ in the following way. Let U be the set of items. For every set $S \in C$ define an agent a_S, whose cost function c_a assigns the value $w(S)$ for every element $s \in S$ and sets the cost of all other elements in U to be infinity. The discounted price function for the agent is shown in figure 2. Here the slope of the second segment is small enough.

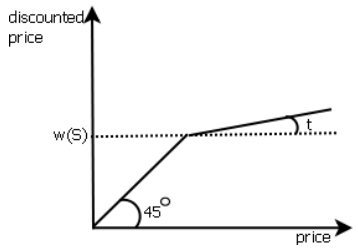

Figure 2: Discount function for agent corresponding to set S

Consider a solution for $\mathcal{I}'$ where we procure at least one item from agent a_S; then we can buy all elements in S from a_S without a significant increase in our payment. So the cost of the optimal solution to $\mathcal{I}$ can be as close to the price of the optimal solution for $\mathcal{I}'$ as we want. By [1, 43], set cover is hard to approximate beyond a factor of $\log n$ unless P = NP. Therefore the discounted reverse auction problem can not be approximated within factor $(1 - o(1)) \log n$ unless P = NP. □

This reduction can be extended to other combinatorial problems in this setting to give logarithmic hardness of approximation for many combinatorial problems. This can be achieved by considering an instance of the problem where we have just one combinatorial object and our task is to allocate it optimally among the agents. For example, for the discounted spanning tree problem we consider the instance when the input graph is itself a tree and we have to optimally allocate its edges among the agents to minimize the total price. Thus, we have the following:

Theorem 3.2.2 *It is hard to approximate discounted edge cover, spanning tree, perfect matching and $s-t$ path within factor of $(1-o(1)) \log n$ on a graph with n vertices unless* P = NP.

3.2.2 Amplification: Hardness for Discounted $s - t$ Path

In this section we consider the discounted $s - t$ path problem between two given vertices s and t. We show that unless P = NP, this problem is hard to approximate within a factor of $O(\log^c n)$ for any fixed constant c. The proof is based on amplification of the result of theorem 3.2.2. This is done by repeatedly applying a transformation σ on the given family of problem instances, which amplifies the approximation factor on every application. Each application of σ also increases the size of the graph but only by a polynomial(in n) factor. We now describe the transformation formally.

Consider the following instance $(G, \mathcal{A}, \mathcal{U})$: We are given a graph $G = (V, E)$, vertices s,t and a set $\mathcal{A}$ of agents. We are also given a collection $\mathcal{U} = \{U_a\}_{a \in \mathcal{A}}$. Here

$U_a \subseteq E$ specifies the set of edges that can be assigned to agent a, i.e., $c_a(e) = 1$ for all $e \in U_a$ and $c_a(e) = +\infty$ otherwise. The discounted price function d_a is such that $d_a(x) = x$ for all $x \leq 1$ and 1 for all $1 < x < +\infty$. Observe that under this assumption, for any set S of edges, $d_a(S)$ has value 1 if $S \subseteq U_a$ and ∞ otherwise. We may assume that the sets U_a for $a \in \mathcal{A}$ are pairwise disjoint, by replacing a single edge that can be assigned to multiple agents by parallel edges and assigning them to each of the agents. In future discussion, we will use $\mathcal{F}$ to denote the family of instances $\{(G, \mathcal{A}, \mathcal{U})\}$.

We define the transformation $\sigma : \mathcal{F} \to \mathcal{F}$ that takes an instance $I = (G, \mathcal{A}, \mathcal{U})$ in $\mathcal{F}$, and generates another instance $I^{\otimes} = (G^{\otimes}, \mathcal{A} \times \mathcal{A}, \mathcal{U}^{\otimes})$ as follows. The graph $G^{\otimes} = (V^{\otimes}, E^{\otimes})$ is constructed from G by replacing each edge $(u, v) \in E$ with a copy of the graph G such that s coincides with u and t coincides with v. Thus any edge $e \in E$ can be identified with a subgraph G^e, of $G^{\otimes}$ that is isomorphic to G. Each $e' \in G^e$ has a cost $c_a(e)$ for each agent a and we define $\rho(e') = e$ and define $\gamma(e')$ to be the edge corresponding to e' in G under this isomorphism. There are $|\mathcal{A}|^2$ agents in the new instance who are indexed by $\mathcal{A} \times \mathcal{A}$. We define the elements of $\mathcal{U}^{\otimes}$ as $U^{\otimes}_{(a_1,a_2)} = \{e' \mid \rho(e') \in U_{a_1} \text{ and } \gamma(e') \in U_{a_2}\}$.

Note that $|E^{\otimes}| = |E|^2$, i.e. the size of instance $I^{\otimes}$ is bounded by a polynomial in the size of I. We define $\sigma(\mathcal{F}) = \{\sigma(I) \mid \forall I \in \mathcal{F}\}$. In lemma 3.2.3, we show that we can amplify that hardness result from theorem 3.2.2 by applying the transformation σ repeatedly.

Lemma 3.2.3 *If $\mathcal{H} = \sigma^r(\mathcal{F})$ is a family of instances for the $s - t$ path problem that is hard to approximate to a factor better than α, then $\sigma(\mathcal{H})$ is hard to approximate within a factor $O(\alpha^2)$.*

Proof: Let $I = (G, s, t, \mathcal{A}, \mathcal{U})$ be an instance in $\mathcal{H}$. Let us begin by making some observations about the structure of an optimal solution for $\sigma(I) = (G^{\otimes}, \mathcal{A}, \mathcal{U}^{\otimes})$.

Claim 3.2.4 *If there is a $s-t$ path of price β in G, then there is a $s-t$ path in $G^{\otimes}$ of price at most β^2.*

Proof: Let $P = e_1, e_2 \dots e_t$ be a path of price β in G. We can construct a $s-t$ path in $G^{\otimes}$ by considering the set of graphs $G^{e_1} \dots G^{e_t}$ and picking the edges corresponding to the edges in P in each of these copies. It can be verified that this gives us a valid path that has price β^2. □

Next we note that the converse is also true.

Claim 3.2.5 *If there is a $s-t$ path of price β^2 in $G^{\otimes}$ then there is a $s-t$ path in G of price at most β.*

Proof: Let P be a path of price β^2 in $G^{\otimes}$. Let $G^{e_1} \dots G^{e_t}$ be the copies of G that have non-empty intersection with P. Two cases may arise. Either the set of edges $\{e_1 \dots e_t\}$ belong to at most β distinct agents in $\mathcal{A}$ or they belong to more than β agents in $\mathcal{A}$. Note that the set of edges $\{e_1 \dots e_t\}$ form a path in G, and in the first case this path has price at most β. In the second case, the price of edges in $P \cap G^{e_i}$ must be less than β for some copy G^{e_i} of graph G. These edges also form a $s-t$ path in G of price at most β. Thus in both cases we can find a path of price at most β in G. □

Using the observations above, if the price of the optimal solution to I is OPT, then the price of the optimal solution to $\sigma(I)$ is OPT^2. Furthermore, if we can approximate the optimal solution to $\sigma(I)$ to within a factor of $o(\alpha^2)$ then we can approximate the optimal solution for I to better than $o(\alpha)$, using the construction in claim 3.2.5. This yields the desired contradiction. □

By theorem 3.2.2, $\mathcal{F}$ is hard to approximate within a factor of $\log n$. Using this as the base case and applying lemma 3.2.3 repeatedly we have the following theorem.

Theorem 3.2.6 *The discounted shortest $s-t$ path problem is hard to approximate within a factor of $O(\log^c n)$ for any fixed constant $c > 0$.*

3.2.3 Reduction: Hardness for Discounted Perfect Matching

In this section we consider the discounted perfect matching problem. We show that unless P = NP, this problem is hard to approximate within a factor of $O(\log^c n)$ for any fixed constant c. The proof is based on a factor preserving reduction from the $s-t$ path problem. We now describe our reduction:

Lemma 3.2.7 *Let A be a β-approximate algorithm for the perfect matching problem, then we can get a β-approximation for the $s-t$ path problem using A as a subroutine.*

Proof: Suppose we are given a graph $G = (V, E)$. Construct an auxiliary graph G^* in the following way: Replace every vertex $v \in V$ by v' and v'' and add an edge connecting them. The price of this edge is zero for every agent. We replace each edge $uv \in E$ with the gadgets shown in figure 3.

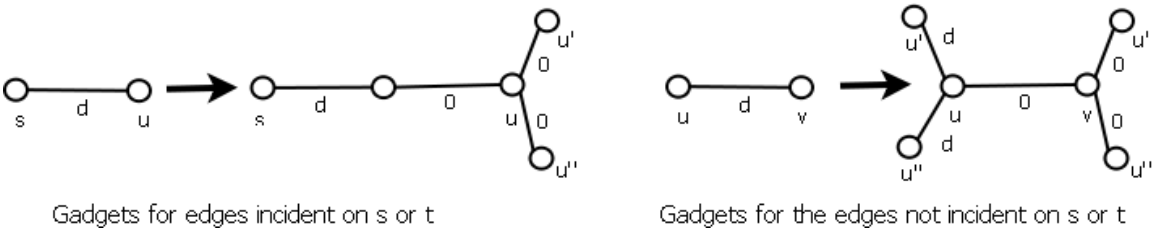

Figure 3: Gadgets

On this graph G^*, use the algorithm A to get the minimum weight matching. Let M be the matching returned. We can interpret M as a $s-t$ path in G in the following way. Let $g(uv)$ be the edges in G^* corresponding to the edge uv for the gadget shown in figure 3. Observe that either one or two edges of every such gadget must belong to M. Let S be the set of edges in G such that two edges in their corresponding gadget belong to M. One can check that every vertex in V is incident with zero or two edges from S, whereas s and t are each incident with exactly one edge in S. Therefore S

consists of an $s-t$ path P_S and some other circuits. Now the circuits in S must have cost zero. This is because if a circuit has positive cost then the cost of the matching can be reduced further by pairing up the vertices in the circuit as shown in figure 4.

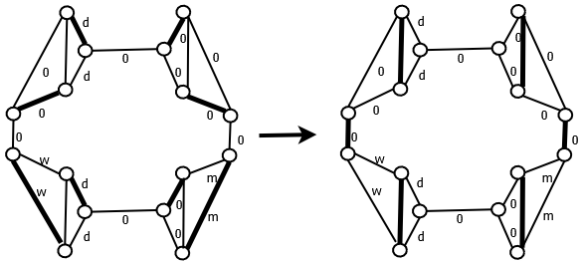

Figure 4: Circuits not involving edges in S should have zero costs

□

Note that the reduction defined in lemma 3.2.7 defines a cost preserving bijection between $s-t$ paths in G to perfect matchings in G^*. Thus, using theorem 3.2.6 we have :

Theorem 3.2.8 *The discounted perfect matching problem is hard to approximate within a factor of* $O(\log^c n)$ *for any fixed constant* $c > 0$.

3.3 Algorithms for Discounted Combinatorial Optimization

In this section, we present approximation algorithms for the four problems defined earlier.

3.3.1 Discounted Edge Cover

We will establish a factor $O(\log n)$ algorithm for the discounted edge cover problem.

Given a discounted edge cover instance, we construct a set cover instance such that: 1) An optimal edge cover corresponds to a set cover with the same cost and 2) A set cover corresponds to an edge cover with a smaller price. For the set cover instance, we apply the greedy algorithm from [31] to get a set cover whose cost is within $\log n$ of the optimal cost. The corresponding edge cover gives a $\log n$ approximation of the

optimum edge cover. We remark that we will have exponentially many sets in the set cover instance that we construct for our problem. To apply the greedy algorithm, we need to show that in each step, the set with the lowest *average cost* can be found in polynomial time.

Now we state our algorithm formally. Consider a set cover instance where we have to cover the set of vertices, V, with $k2^n$ subsets which are indexed by $(a, S) \in \mathcal{A} \times 2^V$. The cost of the set (a, S), denoted by $cost(a, S)$, is defined as the minimum discounted price of an edge cover for the vertices in S that can be built by agent a. For the instance of set cover described above, we apply the greedy algorithm [31] to get a set cover $\mathcal{S}$. Let U_a be the set of vertices covered by sets of the form $(a, S) \in \mathcal{S}$. Let C_a be agent a's optimal discounted edge cover of the vertices in U_a. We output $\{C_a : a \in \mathcal{A}\}$ as our solution.

The correctness of the algorithm follows from the observation that each C_a is a cover of U_a thus their union must form an edge cover for V.

Now we show that the running time of the algorithm is polynomial in k and n. Given U_a, C_a can be found in polynomial time. Thus our algorithm can be implemented in polynomial time, if we can implement the greedy algorithm on our set cover instance efficiently.

Recall that the greedy algorithm from [31] covers the ground set iteratively. Let Q be the set of covered elements at the beginning of a phase. The *average cost* of a set (a, S) is defined as $\alpha_a(S) = cost(a, S)/|S - Q|$. In every iteration the algorithm picks the set with the smallest average cost until all the vertices are covered. To show that this algorithm can be implemented efficiently, we only need to show the following lemma.

Lemma 3.3.1 *For any $Q \subset V$, we can find $min\{cost(a, S)/|S-Q| : (a, S) \in \mathcal{A} \times 2^V\}$ in polynomial time.*

Proof: We can iterate over all choices of agent $a \in \mathcal{A}$, thus the problem boils down to finding $\min\{cost(a, S)/|S - Q| : S \subseteq V\}$ for each $a \in \mathcal{A}$.

For each integer d, if we can find $\min\{cost(a, S) : |S - Q| = d\}$ in polynomial time, then we are done since then we can just search over all the possible sizes of $S - Q$. Unfortunately, it is NP-hard to compute $\min\{cost(a, S) : |S - Q| = d\}$ for all integer d. We will use claim 3.3.2 to circumvent this problem.

Claim 3.3.2 *For any graph $G = (V, E)$ and $Q \subseteq V$ and for any positive integer d, we can find the set (a, S) minimizing $cost(a, S)$ such that $|S - Q|$ is at least d, in polynomial time.*

Proof: To find the desired set we construct a graph $G' = (V', E')$ as follows: Add a set $X \cup Y$ to the set of vertices in G, where $|X| = |Q|$ and $|Y| = |V| - |Q| - d$. Match every vertex in X to a vertex in Q with an edge of cost 0. Connect each vertex in Y to each vertex in V by an edge of very large cost. Set the cost of each edge $e \in E$ as $c_a(e)$. Find the minimum cost edge cover in G'. Let S^* be the set of vertices not adjacent to $X \cup Y$ in such a cover. It is easy to verify that S^* is the desired set. □

By claim 3.3.2 above we can generate a collection of subsets $\{S_i \subseteq V : 1 \leq i \leq n\}$, such that (a, S_i) has the lowest value of $cost(a, S)$ among all sets S which satisfy $|S - Q| \geq i$.

Claim 3.3.3 $min\{cost(a, S)/|S - Q| : S \subseteq V\} = min\{cost(a, S_i)/|S_i - Q| : 1 \leq i \leq n\}$.

Proof: Let S^* be the set that has the minimum average cost with respect to agent a. Suppose $|S^* - Q| = d$. By our choice of S_d, we have $|S_d - Q| \geq d = |S^* - Q|$ and $cost(a, S_d) \leq cost(a, S^*)$. Therefore we have $cost(a, S_d)/|S_d - Q| \leq cost(a, S^*)/|S^* - Q|$, hence they must be equal. □

By iterating over all $a \in \mathcal{A}$, we can find $\min\{cost(a,S)/|S-Q| : (a,S) \in \mathcal{A} \times 2^V\}$ in polynomial time. □

Next we show that the approximation factor of our algorithm is $\log n$. Let OPT_{EC} and OPT_S denote the costs of the optimal solutions for the discounted edge cover instance and the corresponding set cover instance respectively. Let $\{C_a : a \in \mathcal{A}\}$ be the edge cover reported by our algorithm. For all $a \in \mathcal{A}$ let O_a be the set of vertices covered by agent a in the optimal edge cover. The sets $\{(a, O^a) : a \in \mathcal{A}\}$ form a solution for the set cover instance. Therefore $OPT_S \leq OPT_{EC}$.

Since we use the greedy set cover algorithm to approximate OPT_S, we have

$$\sum_{a \in \mathcal{A}} d_a(C_a) \leq (\log n) OPT_S \leq (\log n) OPT_{EC}$$

Thus we have the following theorem.

Theorem 3.3.4 *There is a polynomial time algorithm which finds a $O(\log n)$-approximate solution to the discounted edge cover problem for any graph over n vertices.*

3.3.2 Discounted Spanning Tree

In this section, we study the discounted spanning tree problem and establish an $O(\log n)$ approximation algorithm for this problem.

Let us first consider a simple $O(\log^2 n)$-approximation algorithm. Observe that a spanning tree is an edge cover with the connectivity requirement. If we apply the greedy edge cover algorithm in section 3.3.1, there is no guarantee that we will end up with a connected edge cover. We may get a collection of connected components. We can subsequently contract these components and run the greedy edge cover algorithm again on the contracted graph. We repeat this until there is only one connected component. By this method, we will get a connected edge cover containing a spanning tree.

Now we will analyze the above algorithm. Let OPT_{ST} be the price of the minimum discounted spanning tree. After each execution of the greedy edge cover algorithm,

there is no isolated vertex, hence the contraction decreases the number of vertices by at least a a factor of half, therefore we will have to run the greedy edge cover algorithm at most $O(\log n)$ times. Let OPT^r_{EC} be the price of the minimum edge cover for the graph obtained after the r^{th} contraction and let C_r be the edge cover that we produce for this iteration. Using theorem 3.3.4, the price of C_r is at most $(\log n)OPT^r_{EC}$. It is easy to see that OPT^r_{EC} is at most OPT_{ST} for every r. Hence the price of C_r is bounded by $\log n \cdot OPT_{ST}$. Since there are at most $O(\log n)$ iterations, the price of the spanning tree produced by the above algorithm is bounded by $O(\log^2 n)OPT_{ST}$.

We observe that two main steps in the above algorithm are greedy edge cover and contraction. Intuitively, they are used to satisfy the covering and connectivity requirements respectively. The algorithm proceeds by alternately invoking these subroutines. Based on this observation, our idea to get an $O(\log n)$ approximation algorithm is to apply the following greedy algorithm: rather than apply contraction after each complete execution of the greedy edge cover, we interleave contraction with the iterations of the greedy edge cover algorithm. After each iteration, we modify the graph to coerce our algorithm to get a connected edge cover at the end.

Now we describe our greedy algorithm. For every agent a and subset of vertices S we define $cost(a, S)$ as the cost of the optimal edge cover for S. We define the *average cost* of a set (a, S) as $\alpha^S_a = cost(a, S)/|S|$. The algorithm proceeds in phases and each phase has two steps, **search** and **contraction**. In the r^{th} phase, during the search step we find the set (a_r, S_r) with the lowest average cost and set the *potential* of each vertex $v \in S_r$ as $p(v) = \alpha^{S_r}_{a_r}$. The search step is followed by a contraction step, where we modify the graph by contracting every connected component in the induced subgraph of agent a_r's optimal edge cover for the set S_r. After this we begin the next phase. The algorithm terminates when we have contracted the original graph to a single vertex. For every agent, we find the set of all edges assigned to her across all the search steps declare this as her bundle of assigned edges. Finally remove unnecessary

edges from the set of assigned edges to get a spanning tree.

It is easy to see that we get a connected edge cover at the end of the algorithm, which proves the correctness of the algorithm. To analyze the running time, we observe that there can be at most n phases and by Lemma 3.3.1, each phase can be implemented in polynomial time. Hence the algorithm runs in polynomial time.

Next, we prove that the approximation factor of the algorithm is $O(\log n)$. Let OPT_{ST} be the price of the optimal solution of the discounted spanning tree instance and let $\{T_a : a \in \mathcal{A}\}$ be our solution. Let V' be the set of contracted vertices we produced during the algorithm. Number the elements of V and V' in the order in which they were covered by the algorithm, resolving ties arbitrarily. Suppose $V = \{v_1, ..., v_n\}$ and $V' = \{z_1, ..., z_{n'}\}$. Obviously, $n' \leq n$.

It is easy to verify that $\sum_a d_a(T_a) \leq \sum_i p(v_i) + \sum_j p(z_j)$. Therefore we only need to bound the potentials of the vertices in $V \cup V'$.

Claim 3.3.5 *$p(v_i) \leq \frac{OPT_{ST}}{n-i+1}$ for any $i \in \{1 \cdots n\}$ and $p(z_j) \leq \frac{OPT_{ST}}{n'-j}$ for any $j \in \{1 \cdots n'\}$.*

Proof: For $i \in \{1 \cdots n\}$, suppose v_i is covered in phase r. Let G^r be the underlying graph at the beginning of phase r. Since $v_i, v_{i+1}, ..., v_n$ are not covered before phase r, G^r contains at least $n - i + 1$ vertices. Since the optimal spanning tree can cover the vertices in G^r by a price of OPT_{ST}, by our greedy choice, $p(v_i) \leq OPT_{ST}/(n - i + 1)$.

Similarly, let $1 \leq j \leq n'$ and assume z_j is covered in phase r. Since we should be able to produce $z_{j+1}, z_{j+2}, ..., z_{n'}$ from contraction on vertices of G^r, there are at least $n' - r$ vertices in G^r. Therefore we have $p(z_j) \leq OPT_{ST}/(n' - j)$. □

From the above claim, we have

$$\sum_a d_a(T_a) \leq \sum_{1 \leq i \leq n} \frac{OPT_{ST}}{n-i+1} + \sum_{1 \leq j \leq n'} \frac{OPT_{ST}}{n'-j} \leq (\log n + \log n') OPT_{ST} \leq O(\log n) OPT_{ST}$$

Therefore we have the following theorem:

Theorem 3.3.6 *There is a polynomial time algorithm which finds an $O(\log n)$-approximate solution to the discounted spanning tree problem for any graph with n vertices.*

3.3.3 Discounted $s-t$ Path and Perfect Matching

In sections 3.2.2 and 3.2.3 we showed that unlike edge cover and spanning tree, no polylog-approximate algorithm is likely to exist for discounted $s-t$ path and perfect matching.

Now we describe a simple n-approximate algorithm for discounted $s-t$ path problem. For each edge e, define $w_e = \min_{a\in\mathcal{A}} d_a(c_a(e))$ and for each $s-t$ path P, define its weight $w(P) = \sum_{e\in P} w_e$. Use Dijkstra's algorithm to find a path P_0 with the minimum weight and output it as the solution. Allocate the edges in P_0 as follows: for each edge $e \in P$, we allocate e to the agent a such that $d_a(c_a(e)) = w_e$, with ties broken arbitrarily.

For the analysis, let us define S_a to be the set of edges allocated to agent a in our solution. Since d_a is concave, we have $d_a(S_a) \le \sum_{e\in S_a} d_a(c_a(e))$. Therefore, the total price of our solution is bounded by $\sum_{a\in\mathcal{A}} \sum_{e\in S_a} d_a(c_a(e))$ which is exactly $w(P_0)$. Let OPT be the path chosen in the optimal solution (as an abuse of notation, we also use OPT to denote the optimal value) and OPT_a be the set of edges on the path allocated to agent a. By our choice of P_0 and weight w, we have $w(P_0) \le w(OPT) = \sum_{a\in\mathcal{A}} \sum_{e\in OPT_a} w_e \le \sum_{a\in\mathcal{A}} \sum_{e\in OPT_a} d_a(c_a(e))$. Since d_a is increasing, we have

$$\sum_{e\in OPT_a} d_a(c_a(e)) \le |OPT_a| \cdot d_a(OPT_a) \le n \cdot d_a(OPT_a).$$

Therefore $w(OPT) \le n \sum_{a\in\mathcal{A}} d_a(OPT_a) = n \cdot OPT$. This implies that our algorithm is an n-approximate algorithm.

We apply the same idea for discounted perfect matching problem. Define the weight of a perfect matching M as $w(M) = \sum_{e\in M} w_e$. Use Edmond's algorithm to find a minimum weight perfect matching M_0 for this weight function. For every

$e \in M_0$, allocate it to the agent a such that $c_a(e) = w_e$. By a similar argument as above, we can show that this is a n approximate algorithm.

3.4 Summary

In this chapter, we consider several fundamental submodular combinatorial optimization problems from the computational aspect. We study the class of discounted-price functions which are submodular and succinctly representable. We leave as a concrete open problem to improve our approximation ratios of discounted-price shortest path and perfect matching or enlarge their lower bounds. From a higher level, an interesting future direction would be to investigate the approximabilities of submodular combinatorial optimization problems for other classes of well-motivated succinctly representable functions.

CHAPTER IV

BLACK-BOX REDUCTIONS IN MECHANISM DESIGN

4.1 Introduction

In an *algorithmic mechanism design* problem, we face an optimization problem where the necessary inputs are private valuations held by self-interested agents. The high-level goal of *truthful* mechanisms is to reveal these valuations via the bids of the agents and to optimize the objective simultaneously. In this chapter, we will focus on the objective of social welfare maximization. As usual in computer science, computational tractability is a necessary requirement.

It is well known that the *VCG* mechanism ([10, 22, 52]) which maximizes the social welfare exactly is truthful. However, VCG is not computationally efficient in general. And unfortunately, the simple combination of approximation algorithms and VCG usually fails to preserve truthfulness. This raises the important open question (see [39]) of designing approximation algorithms that are truthful simultaneously.

A major tool that accounts for positive results related to the above question is the *black-box reduction*. The methodology is to apply a given algorithm for the underlying optimization problem as a sub-routine in the design of the mechanism, without any knowledge of the algorithm itself.

In this chapter, we will provide two such reductions for an interesting, broad class of problems called *symmetric single-parameter* problems. Formally, a mechanism design problem (with n agents) is *single-parameter* if each feasible allocation is represented as an n-dimensional real vector $\mathbf{x}$, and each agent i has a private value v_i such that her valuation of allocation $\mathbf{x}$ is given by $v_i x_i$. We further define that a problem is *symmetric* if the set of feasible allocations is closed under permutations:

if $\mathbf{x}$ is feasible, so is $\pi \circ \mathbf{x}$ for any permutation π. Here $\pi \circ \mathbf{x}$ is defined as the vector $(x_{\pi(1)}, ..., x_{\pi(n)})$.

Main Result. Our main result is a black-box reduction that converts *any* approximation algorithm to a truthful mechanism with essentially the *same* approximation factor for the class of symmetric single-parameter problems. In other words, for these problems, *mechanism design is as easy as algorithm design*!

Theorem 4.1.1 *For a symmetric single-parameter mechanism design problem* Π*, suppose we are given an* α*-approximate* ($\alpha > 1$) *algorithm* $\mathcal{A}$ *as a black-box, then for any constant* $\epsilon > 0$*, we can obtain a polynomial time truthful mechanism with approximation factor* $\alpha(1 + \epsilon)$*.*

As an example to which our theorem applies, consider the following Google's TV ad auction problem where an auctioneer is trying to sell m TV ad slots to n advertisers. Assume that there are k TV viewers, and for each TV ad slot we are given the set of viewers who watch that ad slot and we can define a function $f : 2^{[m]} \to [k]$ that for any subset S of ad slots, tells the number of *unique* viewers who will see the ad if the ad is shown on set S. Now suppose that each advertiser has a fixed value (v_i for advertiser i) per unique viewer to whom his ad will be shown. So if a set S_i of ad slots is given to an advertiser i, his total value will be $v_i f(S_i)$. The goal is to design a truthful polynomial time mechanism which maximizes the social welfare $\sum_i v_i f(S_i)$, where S_i is the set of ad slots allocated to advertiser i. Without considering the requirement of truthfulness, Vondrak [53] provides a (1-1/e)-approximate algorithm for social welfare maximization aspect of this problem and it is tight unless P=NP[33]. Applying Vondrak's algorithm as a black-box, Theorem 4.1.1 implies a truthful mechanism with the optimal approximation ratio.

In our reductions, we make no assumption on the black-box algorithm $\mathcal{A}$. In addition, while the black-box algorithm may be randomized, our reduction does not

introduce any further randomization. If the algorithm is deterministic, then our mechanism is deterministically truthful, and if the algorithm is randomized, then our mechanism is universally truthful. Interestingly, this is the first time such reduction is obtained and resolve the conflict between approximability and incentive compatibility for a broad class of problems.

Previous work. There has been a significant amount of work related to black-box reductions in mechanism design. In the single-parameter setting, the first black-box reduction was given by Briest et al. [8]. The authors studied the single-parameter binary optimization problem and they showed that any algorithm which is an FPTAS can be converted to a truthful mechanism that is also an FPTAS. Secondly, Babaioff et al. [3] studied the single-value combinatorial auction problem and they constructed a black-box reduction that converts an algorithm to a truthful mechanism with the approximation factor degraded by a logarithmic factor.

For multi-parameter problems, there is no factor-preserving black-box reduction in general (see [42]). This motivates the study of *truthfulness in expectation*, which is a weaker notion of incentive compatibility. Here, a randomized mechanism is truthful in expectation, if truth telling maximizes an agent's expected payoff. The initial effort in black-box reduction for multi-parameter problems is due to Lavi and Swamy [36], they showed a method to convert a certain type of algorithms called integrality-gap-verifiers to truthful in expectation mechanisms with the same approximation factors. Recently, Dughmi and Roughgarden [14] studied the class of packing problems. Via an elegant black-box reduction and smooth analysis, they showed that if a packing problem admits an FPTAS, then it admits a truthful in expectation mechanism that is an FPTAS as well. At last, Balcan et al.[4] considered black-box reductions from the revenue maximization aspect. By the technique of sample complexity in machine learning, they gave revenue-preserving reductions from truthful mechanism design to

the algorithmic pricing problems.

The previous discussion is about *prior-free* mechanism design. Another important area in algorithmic game theory is the *Bayesian* mechanism design where each agent's valuation is drawn from some publicly known prior distribution. Hartline and Lucier [24] studied this problem in the single-parameter setting. They constructed a clever black-box reduction that converts any non-monotone algorithm into a monotone one without compromising its social welfare. Following this work, Bei and Huang [6] and Hartline et al. [23] independently showed such black-box reductions in the multi-parameter setting as well.

Organization Our constructions are based on the technique of *maximum-in-range*. Here, a maximum-in-range mechanism outputs the allocation maximizing the social welfare over a fixed range of allocations. The range is chosen to balance the following trade-off: A larger range can yield better approximation but require greater computational complexity.

As a warm up, in section 4.3 we show a very simple and efficient construction of the range involving only linear number of queries to the given black-box algorithm and the social welfare maximization within the range requires only linear number of inner product computations. However, the construction suffers from a logarithmic degrade in the approximation factor. To overcome this drawback, we provide a factor-preserving reduction as our main result in section 4.4, which is a more sophisticated construction.

4.2 Preliminaries

In this section, we will outline the basic concepts in mechanism design relevant to our results.

Truthfulness. Let $\mathcal{X}$ be the set of all feasible allocations, and $v_i(\mathbf{x})$ be the private valuation of agent i if allocation $\mathbf{x} \in \mathcal{X}$ is picked. A typical goal of a mechanism is to reveal agents' private valuation functions via their bids and optimize the obtained social welfare simultaneously. Formally, suppose we are given n agents and let $\boldsymbol{v} = (v_1, ..., v_n)$ be the valuation functions reported by the agents. Based on this, a (deterministic) mechanism M will specify an allocation $\mathbf{x}(\boldsymbol{v}) \in \mathcal{X}$ and a payment $\boldsymbol{p}(\boldsymbol{v})$. We say M is *deterministically truthful*(or truthful), if the following conditions hold: for any $i, \boldsymbol{v}_{-i}$ and any v_i, v_i', we have $v_i(\mathbf{x}(v_i, \boldsymbol{v}_{-i})) - p_i(v_i, \boldsymbol{v}_{-i}) \geq v_i(\mathbf{x}(v_i', \boldsymbol{v}_{-i})) - p_i(v_i', \boldsymbol{v}_{-i})$.

Single-parameter mechanism design. In a *single-parameter* mechanism design problem, each allocation is represented as an n-dimensional real vector $\mathbf{x}$ (where n is the number of agents), and each agent i has a private value v_i such that her valuation of allocation $\mathbf{x}$ is given by $v_i x_i$. It is known [38] that for a single-parameter problem, a mechanism is truthful if and only if (1) the allocation rule is *monotone*: suppose $v_i \leq v_i'$, then $x_i(v_i, \boldsymbol{v}_{-i}) \leq x_i(v_i', \boldsymbol{v}_{-i})$; (2) each agent i's payment is determined by $p_i(\boldsymbol{v}) = v_i x_i(v_i, \boldsymbol{v}_{-i}) - \int_0^{v_i} x_i(t, \boldsymbol{v}_{-i}) dt$.

Maximum-in-range mechanisms. The *maximum-in-range* technique is a general approach in the field of mechanism design. It works as follows: The mechanism fixes a range $\mathcal{R}$ of allocations *without* any knowledge of the agents' valuations. Given any $\boldsymbol{v}$, let $\mathbf{x}^*$ =argmax$_{\mathbf{x}\in\mathcal{R}} \sum_j v_j(\mathbf{x})$ and $\mathbf{x}^*_{-i}$ =argmax$_{\mathbf{x}\in\mathcal{R}} \sum_{j\neq i} v_j(\mathbf{x})$ respectively. Now define payment p_i of agent i to be $\sum_{j\neq i} v_j(x^*_{-i}) - \sum_{j\neq i} v_j(x^*)$. It is now not difficult to see that with this payment function, it is in best interest of every agent to report their true valuations, irrespective of what others report. The major challenge in designing maximum-in-range mechanism is to balance between the size of the range and the approximation factor obtained.

4.3 Warm Up: A simple Black-Box Reduction

Recall that a single-parameter mechanism design problem is symmetric if the allocation space $\mathcal{X}$ is closed under permutations: if $\mathbf{x} \in \mathcal{X}$, then $\pi \circ \mathbf{x} = (x_{\pi(1)}, ..., x_{\pi(n)}) \in \mathcal{X}$ for any permutation $\pi \in S_n$. In this section, we provide a very simple black-box reduction for symmetric single-parameter problems that converts any α-approximate algorithm into a truthful mechanism with approximation factor $\alpha \log n$.

Our construction is based on the *maximum-in-range* technique. Given an α-approximate algorithm $\mathcal{A}$, we define a range $\mathcal{R}$ by applying $\mathcal{A}$ as a black-box on some bid vectors fixed in advance. Our mechanism is then maximum-in-range over $\mathcal{R}$.

Construction of the range. Now we describe our construction of the range $\mathcal{R}$. Let $\mathcal{A}$ be an α-approximate algorithm for social welfare maximization in our problem. Consider the following set T of bid vectors $T = \{\boldsymbol{v}^1, \boldsymbol{v}^2, ..., \boldsymbol{v}^n\}$ where $\boldsymbol{v}^j$ is defined as follows:

$$v_i^j = \begin{cases} 1 & \text{For } 1 \leq i \leq j \\ \\ 0 & \text{For } j < i \leq n \end{cases}$$

Let $\mathbf{x}^j$ be the allocation output by $\mathcal{A}$ given the input bid vector $\boldsymbol{v}^j$. Since the allocation space is symmetric, without loss of generality, we may assume that $x_1^j \geq x_2^j \geq ... \geq x_n^j$ for all j. We define our range $\mathcal{R}$ as $\{\pi \circ \mathbf{x}^j : \forall \pi \in S_n, 1 \leq j \leq n\}$. It is easy to see that our construction involves only linear number of value queries.

Given any bid vector, we use $\mathsf{opt}(\boldsymbol{v})$ and $\mathsf{opt}^{\mathcal{R}}(\boldsymbol{v})$ to denote the optimal allocations among $\mathcal{X}$ and $\mathcal{R}$ respectively. As an abuse of notation, we also use them to denote the corresponding optimal social welfare.

Efficiency. Now we show that for any bid vector $\boldsymbol{v}$, we can find $\mathsf{opt}^{\mathcal{R}}(\boldsymbol{v})$ efficiently, although $|\mathcal{R}|$ is exponentially large. Let $\pi^* \in S_n$ be the permutation such that $v_{\pi^*(1)} \geq v_{\pi^*(2)} \geq ... \geq v_{\pi^*(n)}$. Obviously, we have the following:

Claim 4.3.1 $\boldsymbol{v} \cdot (\pi^* \circ \mathbf{x}) \geq \boldsymbol{v} \cdot (\pi \circ \mathbf{x})$ *for all* $\mathbf{x} \in \mathcal{R}$ *and* $\pi \in S_n$.

Therefore, it is easy to see that $\mathsf{opt}^{\mathcal{R}}(\boldsymbol{v}) = \max\{\pi^* \circ \mathbf{x}^1, \pi^* \circ \mathbf{x}^2, ..., \pi^* \circ \mathbf{x}^n\}$, which can be computed by a linear number of inner product computations.

Approximation ratio. Notice that for any $\boldsymbol{v}$, our mechanism outputs $\mathsf{opt}^{\mathcal{R}}(\boldsymbol{v})$. Therefore, we only need to show that $\mathsf{opt}(\boldsymbol{v}) \leq (\alpha \log n)\mathsf{opt}^{\mathcal{R}}(\boldsymbol{v})$ for all $\boldsymbol{v}$.

The high-level intuition of our proof is the following: we will partition the optimal solution into different layers and the one can prove that largest layer provides a $\log n$ approximation of the total social welfare. Then we will show that within each layer, there is an allocation in $\mathcal{R}$ that provides an α-approximation to that particular layer, therefore, the optimal allocation among $\mathcal{R}$ gives an $\alpha \log n$-approximation to $\mathsf{opt}(\boldsymbol{v})$.

Now we formally prove the approximation guarantee. We first need the following:

Definition 1 *Let* $\mathbf{u}, \mathbf{w} \in \mathbb{R}^n_+$. *We say that* $\mathbf{u} \geq_L \mathbf{w}$, *if the following hold:(1)* $u_1 \geq ... \geq u_n$, $w_1 \geq ... \geq w_n$; *(2)* $\sum_{k=1}^n u_k = \sum_{k=1}^n w_k$; *(3) there exists* t *such that* $u_k \geq w_k$ *for all* $k \leq t$ *and* $u_k \leq w_k$ *for all* $k > t$.

The follow lemmas implies that if $\mathbf{u} \geq_L \mathbf{w}$, then the social welfare of $\mathbf{u}$ at least the social welfare of $\mathbf{w}$.

Lemma 4.3.2 *Suppose* $\mathbf{u}, \mathbf{w} \in \mathbb{R}^n_+$ *and* $\mathbf{u} \geq_L \mathbf{w}$. *Let* $\mathbf{x}$ *be such that* $x_1 \geq ... \geq x_n \geq 0$. *Then we have* $\mathbf{u} \cdot \mathbf{x} \geq \mathbf{w} \cdot \mathbf{x}$.

Proof: Let t be such that $u_k \geq w_k$ for all $k \leq t$ and $u_k \leq w_k$ for all $k > t$. Then:

$$\begin{aligned} & \sum_{k=1}^n u_k x_k - \sum_{k=1}^n w_k x_k \\ = & \sum_{k=1}^t (u_k - w_k)x_k + \sum_{k=t+1}^n (u_k - w_k)x_k \\ \geq & \sum_{k=1}^t (u_k - w_k)x_t + \sum_{k=t+1}^n (u_k - w_k)x_t = 0. \end{aligned}$$

Therefore, we have $\mathbf{u} \cdot \mathbf{x} \geq \mathbf{w} \cdot \mathbf{x}$. □

Now we are ready to prove:

Theorem 4.3.3 *For any bid vector $\boldsymbol{v}$, we have $\mathsf{opt}(\boldsymbol{v}) \leq (\alpha \log n)\mathsf{opt}^{\mathcal{R}}(\boldsymbol{v})$.*

Proof: Without loss of generality, we may assume that $v_1 \geq v_2 \geq ... \geq v_n$. For simplicity, we assume $v_{n+1} = 0$. Let $\mathbf{x}^*$ be the optimal allocation. Therefore, we have:

$$\begin{aligned}
& \mathsf{opt}(\boldsymbol{v}) && (11)\\
= & \sum_{i=1}^{n} v_i x_i^* && (12)\\
= & \sum_{j=1}^{n} (v_j - v_{j+1}) \sum_{k=1}^{j} x_k^* && (13)\\
\leq & \alpha \sum_{j=1}^{n} (v_j - v_{j+1}) \sum_{k=1}^{j} x_k^j && (14)\\
\leq & \alpha \sum_{j=1}^{n} (v_j - v_{j+1}) \sum_{k=1}^{j} \frac{j v_k}{\sum_{t=1}^{j} v_t} x_k^j && (15)\\
= & \alpha \sum_{j=1}^{n} \frac{j(v_j - v_{j+1})}{\sum_{t=1}^{j} v_t} \sum_{k=1}^{j} v_k x_k^j && (16)\\
\leq & \big(\alpha \sum_{j=1}^{n} \frac{j(v_j - v_{j+1})}{\sum_{t=1}^{j} v_t}\big)\mathsf{opt}^{\mathcal{R}}(\boldsymbol{v}) && (17)
\end{aligned}$$

Here, equation (12) is from the definition of $\mathbf{x}^*$ and (13) is a rewrite of (12). Since $\mathbf{x}^j$ is the allocation output by $\mathcal{A}$, we have $\sum_{k=1}^{j} x_k^* \leq \alpha \sum_{k=1}^{j} x_k^j$, which implies (14). At last, observe that if we define $\mathbf{u}^j$ as

$$u_k^j = \begin{cases} \dfrac{j v_k}{\sum_{t=1}^{j} v_t} & \text{For } 1 \leq k \leq j \\ \\ 0 & \text{For } j < k \leq n \end{cases}$$

then $\mathbf{u}^j \geq_L \boldsymbol{v}^j$. Therefore, by Lemma 4.3.2, we have (15).

By the previous argument, to complete our proof, we only need to show:

$$\sum_{j=1}^{n} \frac{j(v_j - v_{j+1})}{S_j} \leq \log n.$$

where $S_j = \sum_{t=1}^{j} v_t$. In fact, this is the case because:

$$\begin{aligned}
& \sum_{j=1}^{n} \frac{j(v_j - v_{j+1})}{S_j} \\
= & \sum_{j=1}^{n} \frac{j}{S_j} v_j - \sum_{j=1}^{n} \frac{j}{S_j} v_{j+1} \\
= & \sum_{j=1}^{n} \frac{j}{S_j} v_j - \sum_{j=1}^{n} \frac{j-1}{S_{j-1}} v_j \\
= & \sum_{j=1}^{n} \frac{S_{j-1} - (j-1)v_j}{S_j S_{j-1}} v_j \\
\leq & \sum_{j=1}^{n} \frac{1}{S_j} v_j \\
\leq & \sum_{j=1}^{n} \frac{1}{j} = \log n + o(n)
\end{aligned}$$

This completes our proof. □

The above shows that our simple black-box reduction converts any α-approximate algorithm $\mathcal{A}$ into a truthful mechanism with factor $\alpha \log n$. In fact, our analysis is tight. To overcome this logarithmic degrade, we will provide a more delicate black-box reduction that is factor-preserving in the next section.

4.4 Main Results: A Factor-preserving Reduction

In this section, we will present our main result.

As we mentioned before, the central issue in designing maximum-in-range mechanisms lies in balancing the computational complexity and approximation ratio. Recall that in the previous section, we construct a very simple range among which social welfare maximization requires only linear number of inner-product computations. However, it suffers from a logarithmic degrade in the approximation ratio. In this

section, we will give a more sophisticated construction of the range, which provides us the desired approximation guarantee.

The high-level idea of our construction is the following: Given an algorithm $\mathcal{A}$, we define our range $\mathcal{R}$ by applying $\mathcal{A}$ as a black-box on a carefully chosen collection of *typical bids*. We will show that every bid can be mapped to a typical bid with approximately the same social welfare, hence our mechanism performs at least as well as the algorithm $\mathcal{A}$.

Now we describe our range construction in detail for a given symmetric single-parameter problem Π, black-box algorithm $\mathcal{A}$ and constant $\epsilon > 0$.

4.4.1 Construction of the range

Let $V = \mathbf{R}_+^n$ be the collection of all possible bid vectors. Next we will provide a three-step procedure that chooses a subset $T \subseteq V$ as our collection of *typical bids*.

The first step is *normalization*: By properly reordering the agents and scaling their bids, we only consider the set T_0 of bids where $\boldsymbol{v} \in T_0$ if and only if $1 = v_1 \geq ... \geq v_n$; The second step is *discretization*. In this step, our goal is to obtain a finite set of bid vectors that approximately represent the whole valuation space V. To do this, given any vector $\boldsymbol{v} \in T_0$, we first apply the operation of *tail cutting*: We choose a small value u (e.g. $1/n^M$ for some constant M) and round all the entries smaller than u to 0; then, we discretize the interval $[u, 1]$ by considering $Q = \{\eta^k : k \geq 0\} \cap [u, 1]$ where $\eta < 1$ is a fixed constant. We will round down each of the remaining entries of $\boldsymbol{v}$ after the tail cutting to the closest value in Q. If we do the above for each $\boldsymbol{v} \in T_0$, we obtain a finite set of vectors T_1; The final step is *equalization*. We fix a constant $\beta > 1$ and partition $[n]$ into $\log_\beta n$ groups. For each vector in T_1, we equalize its entries within each group by setting them to be the value of the largest entry in the group. We then obtain the set of vectors T, and each vector in T is called a *typical bid*.

Now we provide the detailed description. In the following, we fix constants $\beta > 1$ and $\eta < 1$ such that $\frac{\beta}{\eta} = 1 + \epsilon/2$.

1: **Normalization.** Let $T_0 = \{\boldsymbol{v} : 1 = v_1 \geq ... \geq v_n\}$;

2: **Discretizing.** Let $Q = \{\eta^k : 0 \leq k \leq \lceil \log_{1/\eta}(n^M) \rceil\}$ where $M \geq \log_2 \frac{8}{\epsilon}$ is a constant. For any real value z, define $\lfloor z \rfloor_\eta = \eta^{\lceil \log_{1/\eta} z \rceil} \in Q$. Then we define a function $D : T_0 \mapsto T_0$ as follows: for each $\boldsymbol{v} \in T_0$ and for each i, define

$$D(\boldsymbol{v})_i = \begin{cases} \lfloor v_i \rfloor_\eta & v_i \geq u = \frac{1}{n^M} \\ 0 & \text{otherwise} \end{cases}$$

Let $T_1 = D(T_0)$;

3: **Equalization.** Let $n_k = \lfloor \beta^k \rfloor$ where $\beta > 1$ is a fixed constant and $0 \leq k \leq \lfloor \log_\beta n \rfloor$. Define a function $E : T_1 \to T_1$ as follows: for each $\boldsymbol{v} \in T_1$ and $1 \leq i \leq n$, $E(\boldsymbol{v})_i$ is set to be v_{n_k} when $n_k \leq i < n_{k+1}$. At last, let $T = E(T_1)$.

For a bid vector $\boldsymbol{v}$, let $\mathbf{x}^{\mathcal{A}}(\boldsymbol{v})$ be the allocation obtained by applying algorithm $\mathcal{A}$ on $\boldsymbol{v}$. Since the allocation space is closed under permutations, we may assume $\mathbf{x}^{\mathcal{A}}(\boldsymbol{v})_1 \geq \mathbf{x}^{\mathcal{A}}(\boldsymbol{v})_2 \geq ... \geq \mathbf{x}^{\mathcal{A}}(\boldsymbol{v})_n$. At last, let $\mathcal{R}_0 = \{\mathbf{x}^{\mathcal{A}}(\boldsymbol{v}) : \boldsymbol{v} \in T\}$ and we finally define our range as $\mathcal{R} = \{\pi \circ \mathbf{x} : \mathbf{x} \in \mathcal{R}_0, \pi \in \Pi_n\}$ where Π_n consists of all permutations over n elements.

4.4.2 Analysis

Now we analyze the performance of our mechanism. Since the mechanism is maximum-in-range, it is truthful. We will show that it has polynomial running time and an approximation factor of $\alpha(1 + \epsilon)$.

4.4.2.1 Running time

In this section, we show that the social welfare maximization over $\mathcal{R}$ is solvable within polynomial time, hence our maximum-in-range mechanism is efficient.

First of all, we show that the $|\mathcal{R}_0|$ is polynomial in n. We only need to prove the following lemma:

Lemma 4.4.1 $|\mathcal{R}_0| \leq |T| \leq n^{1/\log_2 \beta + M/\log_2(1/\eta)}$.

Proof: The first inequality follows from the definition of $\mathcal{R}_0$. Now we prove the second one. Observe that for each vector $\boldsymbol{v}$ in T_1, $E(\boldsymbol{v})$ is uniquely determined by the values $\{v_{n_k} : 0 \leq k \leq \lfloor \log_\beta n \rfloor\} \subseteq Q \bigcup \{0\}$. Moreover, we have that $v_{n_{k-1}} \geq v_{n_k}$ for all k. Therefore, let H be the class of non-increasing functions from $\{0, 1, ..., \lfloor \log_\beta n \rfloor\}$ to $Q \bigcup \{0\}$, thus $|T| \leq |H|$. Since $|Q| = \lceil \log_{1/\eta}(n^M) \rceil$, It is not difficult to see,

$$|H| \leq \binom{\lfloor \log_\beta n \rfloor + \lceil \log_{1/\eta}(n^M) \rceil}{\lceil \log_{1/\eta}(n^M) \rceil} \leq 2^{\lfloor \log_\beta n \rfloor + \lceil \log_{1/\eta}(n^M) \rceil} \leq n^{1/\log_2 \beta + M/\log_2(1/\eta)}.$$

□

Now we are ready to prove the running time guarantee. Let $\mathsf{opt}_{\mathcal{R}}(\boldsymbol{v})$ be the allocation maximizes the social welfare over $\mathcal{R}$ for the given bid vector $\boldsymbol{v}$. Let σ be the permutation such that $v_{\sigma(1)} \geq ... \geq v_{\sigma(n)}$. Obviously, for each $\mathbf{x} \in \mathcal{R}_0$, we have $\boldsymbol{v} \cdot (\sigma^{-1} \circ \mathbf{x}) \geq \boldsymbol{v} \cdot (\pi \circ \mathbf{x})$ for all permutation π. Therefore, $\mathsf{opt}_{\mathcal{R}}(\boldsymbol{v}) \in \{\sigma^{-1} \circ \mathbf{x} : \mathbf{x} \in \mathcal{R}_0\}$. By Lemma 4.4.1, $|\{\sigma^{-1} \circ \mathbf{x} : \mathbf{x} \in \mathcal{R}_0\}| = |\mathcal{R}_0| \leq n^{1/\log_2 \beta + M/\log_2 \eta}$, this implies that $\mathsf{opt}_{\mathcal{R}}(\boldsymbol{v})$ can be found in polynomial time.

4.4.2.2 Approximation factor

In this section, we show that the approximation factor of our mechanism is $\alpha(1+\epsilon)$. Given any bid vector $\boldsymbol{v}$, by reordering and scaling properly, we may assume $\boldsymbol{v} \in T_0$, then we consider the typical bid $E(D(\boldsymbol{v}))$. We show that for any sorted allocation $\mathbf{x}$, the social welfare $\boldsymbol{v} \cdot \mathbf{x}$ is $(1+\epsilon)$-approximated by $E(D(\boldsymbol{v})) \cdot \mathbf{x}$, hence an α-approximate solution for social welfare maximization with respect to $E(D(\boldsymbol{v}))$ is an $\alpha(1+\epsilon)$-approximate solution for $\boldsymbol{v}$. This proves the desired approximation guarantee.

Now we provide the detail. We first show that by considering $D(\boldsymbol{v})$ instead of $\boldsymbol{v} \in T_0$, the social welfare is rounded down by at most a factor of $\eta(1 - \epsilon/4)$.

Lemma 4.4.2 *For any $\boldsymbol{v} \in T_0$ and any allocation $\mathbf{x}$ such that $x_1 \geq ... \geq x_n$, we have $D(\boldsymbol{v}) \cdot \mathbf{x} \leq \boldsymbol{v} \cdot \mathbf{x} \leq \frac{1}{\eta(1-\epsilon/4)} D(\boldsymbol{v}) \cdot \mathbf{x}$.*

Proof: The first inequality holds by definition. Now we prove the second one. We first show that the social welfare affected by "tail cutting" is bounded by a fraction of $\epsilon/4$.

Claim 4.4.3 $\sum_{i:v_i \geq 1/n^M} v_i x_i \geq (1-\epsilon/4) \sum_{i=1}^n v_i x_i$.

Proof: For the ease of notation, we let $A = \sum_{i=1}^n v_i x_i$ and $B = \sum_{i:v_i \geq 1/n^M} v_i x_i$. Thus

$$A = B + \sum_{i:v_i < 1/n^M} v_i x_i \leq B + \frac{1}{n^M} \sum_{i:v_i < 1/n^M} x_i.$$

Since $x_1 \geq ... \geq x_n$ and $1 = v_1 \geq ... \geq v_n$, we have

$$\frac{1}{n} \sum_{i=1}^n x_i \leq \sum_{i=1}^n \frac{\sum_{j=1}^n v_j}{n} x_i \leq \sum_{i=1}^n v_i x_i = A.$$

The last inequality holds because of rearrangement inequality. Therefore, we have

$$A \leq B + \frac{1}{n^{M-1}} \big(\frac{1}{n} \sum_{i:v_i < 1/n^M} x_i\big) \leq B + \frac{1}{n^{M-1}} \big(\frac{1}{n} \sum_{i=1}^n x_i\big) \leq B + \frac{1}{n^{M-1}} A \leq B + \frac{\epsilon}{4} A.$$

Hence we have $B \geq (1-\epsilon/4)A$. □

Let $\boldsymbol{v}' = D(\boldsymbol{v})$, it is easy to see:

$$\boldsymbol{v}' \cdot \mathbf{x} = \sum_{i:v_i \geq 1/n^M} v_i' x_i \geq \eta \sum_{i:v_i \geq 1/n^M} v_i x_i \geq \eta(1 - \frac{\epsilon}{4}) \boldsymbol{v} \cdot \mathbf{x}$$

. □

Secondly, we show that the social welfare increases by at most a factor of β by considering $E(\boldsymbol{v})$ instead of $\boldsymbol{v}$ for any $\boldsymbol{v} \in T_1$.

Lemma 4.4.4 *For any $\boldsymbol{v} \in T_1$ and any allocation $\mathbf{x}$ such that $x_1 \geq \cdots \geq x_n$, we have $\boldsymbol{v} \cdot \mathbf{x} \leq E(\boldsymbol{v}) \cdot \mathbf{x} \leq \beta \boldsymbol{v} \cdot \mathbf{x}$.*

Proof: The first inequality is implied by the definition of E. We will prove the second one. Let $\boldsymbol{v}' = E(\boldsymbol{v})$ and $L = \lceil \log_{1/\eta} n \rceil$. Since $\boldsymbol{v}, \boldsymbol{v}' \in T_1$, we have that $v_i, v_i' \in Q$ for each i. Thus, if we let $l_i = \log_\eta v_i$ and $l_i' = \log_\eta v_i'$ respectively for each i, then l_i's and l_i''s are non-decreasing sequences. By our construction, it is easy to see: (1) for all $1 \le i \le n$, $l_i' \le l_i$; (2) for all $l \in [0, L]$, $|\{i : l_i' \le l\}| \le \beta\, |\{i : l_i \le l\}|$. Since $x_1 \ge ... \ge x_n$, we have the following:

Claim 4.4.5 *For any $l \in [0, L]$ and $1 \le i \le n$, $\sum_{i:l_i' \le l} x_i \le \beta \sum_{i:l_i \le l} x_i$.*

Observe that if we define $W_l = \eta^l$ for $0 \le l \le L$ and $W_{L+1} = 0$, then

$$\sum_{i=1}^{n} x_i v_i = \sum_{i=1}^{n} x_i W_{l_i} = \sum_{i=1}^{n} x_i \sum_{l=l_i}^{L} (W_l - W_{l+1}) = \sum_{l=0}^{L} (W_l - W_{l+1}) \sum_{i:l_i \le l} x_i \ .$$

Similarly, we have $\sum_{i=1}^{n} x_i v_i' = \sum_{l=0}^{L} (W_l - W_{l+1}) \sum_{i:l_i' \le l} x_i$. By Lemma 4.4.5, we have $\boldsymbol{v}' \cdot \mathbf{x} \le \beta \boldsymbol{v} \cdot \mathbf{x}$. □

By Lemma 4.4.2 and Lemma 4.4.4, we have the following:

Corollary 4.4.6 *For any $\boldsymbol{v} \in T_0$ and any allocation $\mathbf{x}$ such that $x_1 \ge \cdots \ge x_n$, we have: $\eta(1 - \epsilon/4)\boldsymbol{v} \cdot \mathbf{x} \le E(D(\boldsymbol{v})) \cdot \mathbf{x} \le \beta \boldsymbol{v} \cdot \mathbf{x}$.*

Now we prove the approximation guarantee of our mechanism. Given any bid vector $\boldsymbol{v}$, without loss of generality, we may assume $\boldsymbol{v} \in T_0$. Let $\boldsymbol{z}^*$ be the optimal solution of social welfare maximization for $\boldsymbol{v}$ and $\mathbf{x}^*$ be the solution output by our mechanism. In addition, let $\boldsymbol{y}^*$ be the optimal solution for the typical bid $E(D(\boldsymbol{v}))$. Then, by Corollary 4.4.6, we have

$$\boldsymbol{v} \cdot \mathbf{x}^* \ge \frac{1}{\beta} E(D(\boldsymbol{v})) \cdot \mathbf{x}^* \ . \tag{18}$$

Since our algorithm is maximum-in-range, allocation $\mathbf{x}^*$ is at least as good as the allocation by algorithm $\mathcal{A}$ with respect to typical bid vector $E(D(\boldsymbol{v}))$. Hence, we have

$$E(D(\boldsymbol{v})) \cdot \mathbf{x}^* \ge \frac{1}{\alpha} E(D(\boldsymbol{v})) \cdot \boldsymbol{y}^* \ . \tag{19}$$

Further, by optimality of $\boldsymbol{y}^*$ and Corollary 4.4.6, we have

$$E(D(\boldsymbol{v})) \cdot \boldsymbol{y}^* \geq E(D(\boldsymbol{v})) \cdot \boldsymbol{z}^* \geq \eta \left(1 - \frac{\epsilon}{4}\right) \boldsymbol{v} \cdot \boldsymbol{z}^* \ . \tag{20}$$

Since we choose β and η such that $\beta/\eta = 1 + \epsilon/2$, by equation 18, 19 and 20, we have $\boldsymbol{v} \cdot \boldsymbol{z}^* \leq \alpha(1+\epsilon/2)/(1-\epsilon/4)\boldsymbol{v} \cdot \mathbf{x}^* \leq \alpha(1+\epsilon)\boldsymbol{v} \cdot \mathbf{x}^*$. This completes our analysis.

4.5 Summary

In this chapter, we provide two black-box reductions for the class of symmetric single-parameter problems. Arguably, this is the first time such results are shown in the prior-free setting for a broad class of problems. An interesting and important future direction is to either generalize our result to multi-parameter problems, or establish lower bounds on the approximabilities for truthful mechanisms, or at least, maximum-in-range mechanisms.

CHAPTER V

OPTIMAL AUCTION DESIGN

5.1 Introduction

Optimal auction design is an important subject that has been heavily studied in both economics and theoretical computer science. Among the accomplished research in this area, a solid part is focused on *single-item auction*, which serves as a basic that provides insight to other more complicated problems. In the seminar paper [38], Myerson gave a complete characterization of the optimal single-item auction in the setting where bidders' valuations are drawn from independent distributions. However, the design of optimal auction with correlated bidders was left open.

From the economics aspect, a natural attempt for solving this problem is to generalize Myerson's characterization. Unfortunately, most results obtained via this approach are for restricted special cases, see [35] for a survey. On the other hand, from a computer science aspect, two research directions (see [13]) were suggested.

The first one is the introduction of approximation algorithms into optimal auction design. In other word, instead of providing a characterization of the optimal auction, which might not even exist, one would look for efficient algorithms that guarantee the approximate optimality.

Along this direction, two computational models were considered-the *explicit model* [41] and the *oracle model* [44]. In the explicit model, the running time of an algorithm has to be polynomial in the support size of the distribution. However, in the oracle model, the algorithm is only allowed to make polynomially many queries to an oracle that returns the conditional distribution of a set of bidders given the values of the remaining ones. Although several positive and negative results have been discovered in

both models [13, 41, 44, 45], understanding the approximability of the optimal auction remains as a major challenge.

The second direction suggested is to relax the solution concept to *truthfulness-in-expectation*. One advantage of such relaxation is that the optimal truthful-in-expectation auction can be described as a linear program [13] whose size is polynomial in the support of the distribution, hence can be computed efficiently in the explicit model.

Based on this observation, Dobzinski et.al. [13] studied a class of truthful-in-expectation mechanisms called *k-lookahead*. To be precise, for any fixed constant k, the k-lookahead mechanism runs the linear program among the k bidders with the highest bids, conditioning on the remaining bidders. Since k is a constant, the linear program can be solved efficiently in the oracle model.

In, [13], the authors showed that the k-lookahead mechanism has approximation ratio $\frac{3k-1}{2k-1}$. As usual in computer science, improving this approximation ratio would be an important issue in this direction. Furthermore, a question that is of theoretical interest itself is the task of evaluating the gap between truthful-in-expectation and deterministically truthful mechanisms. Obviously, one would expect truthful-in-expectation mechanisms to achieve more revenue than the deterministic ones. Dobzinski et.al. showed that the gap is at most a factor of 5/3 by an elegant derandomization of the 2-lookahead mechanism. Closing the gap further requires either better truthful-in-expectation mechanisms that can be derandomized, or simply tighter analysis of the 2-lookahead mechanism.

Our results In this chapter, we contribute to both research directions mentioned earlier by providing more delicate analysis of the k-lookahead mechanisms in the oracle model. We show that the approximation ratio of k-lookahead mechanism is at least $\frac{e^{1-1/k}}{1+e^{1-1/k}}$, which improves the ratio given in [13]. In particular, our result implies

that 2-lookahead mechanism is at least $\frac{\sqrt{e}}{1+\sqrt{e}}$-approximate and interestingly, we prove that our analysis is tight by showing an example in which 2-lookahead mechanism obtains exactly $\frac{\sqrt{e}}{1+\sqrt{e}}$ fraction of the optimal revenue.

Our analysis is based on the clever idea from [13] of comparing the revenue obtained by k-lookahead mechanism to the *t-fixed-price* and *t-pivot* auctions. The novelty of our approach is that instead of picking only one *threshold* t, we consider a series of thresholds $t_1, ..., t_m$ and choose the best series. Apparently, our analysis will lead to better ratio but become more complicated. Therefore, new idea and technique will be introduced for our analysis.

Related work The celebrated result of Myerson [38] initiated the research in optimal auction design. In the paper, Myerson studied the case of single-item auction with independent bidders. An important open question left was the optimal auction for correlated bidders.

As we mentioned, an economic approach of studying the single-item auction is via characterizing the revenue maximizing mechanism for special distributions, see [35]. One exception is [11] by Cremer and McLean where they relax the *individually rational* constraint and obtain mechanisms that extract the full social welfare.

From the computer science stand point, Ronen [44] gave the first efficient mechanism in the oracle model called 1-lookahead that 2-approximates the optimal revenue. In [45], Ronen and Saberi further proved that no deterministic efficient *ascending auction* can do better than $\frac{3}{4}$. On the other hand, in the explicit model, Papadimitiou and Pierrakos [41] showed that although the optimal auction for two bidders can be computed efficiently, it is NP-hard to do so for more than three bidders.

Most recently, Dobzinski, Fu and Kleinberg [13] generalize the 1-lookahead mechanism to k-lookahead, which is truthful in expectation and efficient in the oracle model. They prove that the approximation ratio of k-lookahead is $\frac{3k-1}{2k-1}$. They further show

that 2-lookahead can be derandomized, which implies a gap of at most 5/3 between deterministic and truthful-in-expectation mechanisms.

Organization We first provide some preparation and terminology in section 5.2. We present our analysis of k-lookahead mechanism in section 5.3 and the tightness example for 2-lookahead will be described in section 5.4.

5.2 Preliminary

In this section, we formally define our problem and provide some useful facts that will be needed in the future discussion.

In a single-item auction, a seller wishes to sell one item to a group of n self-interested bidders. Each bidder has a private valuation $v_i \in \mathbb{R}^+$. We assume that there is a publicly known distribution $\mathcal{D}$ on the valuation space of the bidders. In this chapter, we make no assumption on the distribution. In particular, bidders' valuations could be correlated. Since we only consider truthful mechanisms in this paper, we will equalize the notions of *bid* and *valuation*.

An auction M is a mechanism that takes a bid vector $\boldsymbol{v}$ and then decides who wins the item and for what price. We use $(\mathbf{x}, \boldsymbol{p})$ to denote the allocation and payment where $x_i(\boldsymbol{v})$ is the probability that bidder i gets the item and $p_i(\boldsymbol{v})$ is her expected payment. Here, the goal of each bidder i is to maximize her own *utility* defined as $x_i v_i - p_i$.

A mechanism is *deterministically truthful* if reporting the true valuation is a dominant strategy for each agent and we say that a randomized mechanism is *universally truthful* if the mechanism is a probability distribution over deterministically truthful mechanisms. At last, *truthful-in-expectation* is a weaker notion in which an agent maximizes her expected utility by being truthful. It is easy to see that every deterministically truthful mechanism is universally truthful and every universally truthful mechanism is truthful in expectation.

In this chapter, we are interested in designing truthful-in-expectation mechanisms. From now on, without particular specification, we will simply say a mechanism is *truthful* if it is truthful-in-expectation and an *optimal auction* is referred to a truthful-in-expectation mechanism that maximizes the seller's expected profit

$$E_{\mathcal{D}}[M] = E_{\boldsymbol{v}\sim\mathcal{D}}(\sum_{i=1}^{n} p_i(\boldsymbol{v}))$$

on input distribution $\mathcal{D}$.

An useful observation is that the optimal auction can be described by the following linear program [13].

$$\begin{aligned}
& \max \sum_{\boldsymbol{v}\sim\mathcal{D}} Pr_{\mathcal{D}}(\boldsymbol{v}) \sum_i p_i(\boldsymbol{v}) \\
s.t. \quad & \sum_{\boldsymbol{v},i} x_i(\boldsymbol{v}) \le 1, \qquad \forall \boldsymbol{v}, i; \\
& x_i(\boldsymbol{v})v_i - p_i(\boldsymbol{v}) \ge x_i(v_i', v_{-i})v_i - p_i(v_i', v_{-i}), \qquad \forall i, \boldsymbol{v}, v_i'; \\
& x_i(\boldsymbol{v}) \ge 0, p_i(\boldsymbol{v}) \ge 0 \qquad \forall i, \boldsymbol{v}; \\
& p_i(0, v_{-i}) = 0, \qquad \forall i, v_{-i}.
\end{aligned}$$

Here, v_{-i} be the valuation vector of all bidders except bidder i, i.e. $v_{-i} = (v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n)$ and (v_i', v_{-i}) denotes the vector $(v_1, \cdots, v_{i-1}, v_i', v_{i+1}, \cdots, v_n)$. As one can see, the size of the linear program is polynomial in the support size of the distribution, which implies that the optimal auction can be computed efficiently in the explicit model. However, the linear program is not generally efficient in the oracle model, unless the number of bidders is a constant. This motivates the study of k-lookahead mechanisms [13, 44].

In a k-lookahead mechanism, we find the k bidders with the highest values. Without lose of generality, we assume these k bidders are $v_1, v_2, \cdots, v_k$ and denote the set of these k bidders by K. Next we get the conditional distribution D_K on

$v_i' \geq \max\{v_j | j \notin K\} (\forall i \in K)$ and $v_{k+1}, \cdots, v_n$ is fixed. Then we reject the bidders not in K and use the mentioned linear program for distribution D_K to get the allocation vector $\mathbf{x}_K$ and payment vector $\boldsymbol{p}_K$.

In this paper, we will investigate the approximation ratio of the k-lookahead mechanism. Here, we say an auction M is a c-approximation mechanism if $\frac{E_{\mathcal{D}}[M]}{E_{\mathcal{D}}[OPT]} \geq c$ where OPT is the revenue-maximizing valid auction on distribution $\mathcal{D}$.

Finally, the following theorem provides a characterization of deterministic mechanisms for single item auctions, which will be useful in the analysis of 2-lookahead mechanism.

Theorem 5.2.1 *[38] A deterministic mechanism, with allocation and payment rule $\boldsymbol{q}, \boldsymbol{p}$ respectively, is truthful if and only if for each bidder i and each v_{-i}, the following conditions hold:*

1. Monotone Allocation*: $q_i(v_i, v_{-i}) \leq q_i(v_i', v_{-i})$ for all $v_i \leq v_i'$;*
2. Threshold Payment*: There exists a threshold $t_i(v\_i)$ such that $p_i(v_i, v_{-i}) = t_i(v_{-i}) \cdot q_i(v_i, v_{-i})$.*

5.3 The Approximation Ratio

In this section, we present our main result. From now on, we fix a constant k and let K be the agents with the highest k bids. Let D_K be the conditional distribution of bidders in K conditioned on the remaining bidders. We show that the approximation ratio of k-lookahead mechanism is at least $\frac{e^{1-1/k}}{1+e^{1-1/k}}$.

Our high-level idea is to partition the optimal revenue into different components. Then we design several auctions that only sell the item in K and each of them approximately realizes part of the components. The revenues of these auctions provide a lower bound on the revenue of k-lookahead since it is the optimal auction that only sells the item to bidders in K.

In the following, we always assume that the optimal revenue is 1. Now we consider the expected revenue of the k-lookahead auction. As we mentioned before, we first partition the optimal revenue into four parts.

Definition 2 *Fix the optimal auction, for any $t > 1$, we define $L(t), \tilde{L}(t), M(t), H(t)$ as follows:*

1. *$L(t)$:the expected revenue from bidders in $N \backslash K$ for instances where no bidder in K has value at least $t \cdot v_{k+1}$.*

2. *$\tilde{L}(t)$:the expected revenue from bidders in $N \backslash K$ for instances where there are some bidders in K whose valuations are at least $t \cdot v_{k+1}$.*

3. *$M(t)$:the expected revenue from bidders in K for instances where no bidder in K has value at least $t \cdot v_{k+1}$.*

4. *$H(t)$:the expected revenue from bidders in K for instances where there are some bidders in K whose valuations are at least $t \cdot v_{k+1}$.*

Let the expected revenue from K in the optimal auction be $\alpha (\alpha \leq 1)$. By our definition, $M(t) + H(t) = \alpha$ and $L(t) + \tilde{L}(t) = 1 - \alpha$ for all $t \geq 1$.

Lemma 5.3.1 *The expected revenue of k-lookahead auction is at least α.*

Proof: Consider the following auction: If the optimal auction sells the item to bidder i in K with probability x_i and p_i, we still sell the item to i with probability x_i and ask for a payment p_i. Otherwise no one gets the item. This mechanism might not be truthful because it is possible that some bidder in $N \backslash K$ raises her bid so that she becomes a bidder in K and has a chance to get the item. To make this mechanism truthful, we raise the expected payment of each bidder i by $\max\{0, (v_{k+1} - p_i(v_{k+1}, v_{-i})) \cdot \frac{x_i(v_{k+1}, v_{-i})}{x_i(v)}\}$. This is then a truthful mechanism with expected revenue

at least α. Furthermore, one can see that the mechanism only sells the item to bidders in K, therefore, the expected revenue of k-lookahead auction is at least α. $\square$

The above lemma provides a lower bound on the revenue of k-lookahead related to the components of M and H in the optimal auction. To get more such bounds, we need the following auctions first introduced by Dobzinski, Fu and Kleinberg [13]. Suppose there is a thresholds $t \geq 1$. Without lose of generality, we assume v_{k+1} is the highest valuation not in K:

1. t-Fixed Price Auction: Select one bidder j from K uniformly at random. If there is any bidder in $K \backslash \{j\}$ has valuation at least $t \cdot v_{k+1}$ then he gets the item with payment $t \cdot v_{k+1}$. If there are several bidders satisfy this condition, break ties arbitrary. Otherwise, we allocate the item to bidder j with a payment v_{k+1}.

2. t-Pivot Auction: Select one bidder j from K uniformly at random. If there is any bidder in $K \backslash \{j\}$ has valuation greater than $t \cdot v_{k+1}$, we choose the bidder i with the smallest index. We run the *k-lookahead auction* on the conditional distribution D'_k that fixed the valuations of bidders not in K, required $v'_l \geq v_{k+1} (l \in K)$ and $v'_i \geq t \cdot v_{k+1}$. Otherwise, we allocate the item to bidder j with a payment v_{k+1}.

It is easy to verify that t-Fixed Price Auction is truthful. To check t-Pivot Auction is truthful, the only case we should be careful is that some bidder i raises her valuation and let the mechanism run the k-lookahead auction. However, bidder i must be the only bidder whose valuation is no less than $t \cdot v_{k+1}$ in this case. Therefore k-lookahead auction runs under the conditional distribution that $v_i > t \cdot v_{k+1}$. As a result, her payment must be at least $t \cdot v_{k+1}$, which exceeds her actual valuation. Therefore, t-Pivot auction is truthful.

In the following, we will choose a series of s threshold values $t_1 < t_2 < \cdots < t_s$ (whose values will be determined later) and relate the revenue of each t_i-Fixed Price

Auction and t_i-Pivot Auction to the four components of the optimal revenue defined earlier.

To be precise, assume that $t_0 = 1$ and we define $M_i = M(t_i) - M(t_{i-1}) \geq 0$ which is the revenue from K in the optimal auction when the highest valuation is in $[t_{i-1} \cdot v_{k+1}, t_i \cdot v_{k+1})$.

Lemma 5.3.2 *The expected revenue of t_i-Fixed Price Auction is at least $P_i = L(t_i) + \sum_{j=1}^{i} \frac{M_j}{t_j} + (\frac{k-1}{k} t_i + \frac{1}{k})(\tilde{L}(t_i) + \sum_{j=i+1}^{s} \frac{M_j}{t_j})$.*

Proof: We consider two cases. In the first case, there is no bidder in K whose valuation is greater or equal than $t_i \cdot v_{k+1}$. So the auction allocates the item to the selected bidder j with payment v_{k+1}. The corresponding expected revenue in the optimal auction is $L(t_i) + \sum_{j=1}^{i} M_j$. From the definition of M_i, the revenue of our auction is at least $L(t_i) + \sum_{j=1}^{i} \frac{M_j}{t_j}$.

In the second case, there are some bidders whose valuations are at least $t_i \cdot v_{k+1}$. In our auction, the auction will obtain $t_i \cdot v_{k+1}$ with probability at least $\frac{k-1}{k}$. Otherwise the auction will obtain at least v_{k+1}. Therefore the expected revenue of this auction is at least $(\frac{k-1}{k} t_i + \frac{1}{k})\tilde{L}(t_i)$ when the optimal auction allocates the item to bidders not in K. At the same time, the expected revenue of this auction is at least $(\frac{k-1}{k} t_i + \frac{1}{k}) \sum_{j=i+1}^{s} \frac{M_j}{t_j}$ when the optimal auction allocates the item to K.

From all discussion above, the expected revenue of t_i-Fixed Price Auction is at least $P_i = L(t_i) + \sum_{j=1}^{i} \frac{M_j}{t_j} + (\frac{k-1}{k} t_i + \frac{1}{k})\tilde{L}(t_i) + (\frac{k-1}{k} t_i + \frac{1}{k}) \sum_{j=i+1}^{s} \frac{M_j}{t_j}$. □

Similarly, we can prove the following:

Lemma 5.3.3 *The expected revenue of t_i-Pivot Auction is at least $Q_i = L(t_i) + \sum_{j=1}^{i} \frac{M_j}{t_j} + \frac{k-1}{k} H(t_i) + \frac{1}{k}(\tilde{L}(t_i) + \sum_{j=i+1}^{s} \frac{M_j}{t_j})$.*

Let $R_i = \max\{P_i, Q_i\}$ and we can see that $\max_{1 \leq i \leq s} R_i$ is a lower bound on the revenue of k-lookahead. From the above lemma, this lower bound is explicitly related

to the components M, H, L and $\tilde{L}$. In the following, we will choose s large enough and $t_1, ..., t_s$ appropriately to obtain a lower bound on $\max_{1\le i\le s} R_i$ that is only related to α. Together with Lemma 5.3.1, we will get the desired approximation ratio.

Now we prove this lower bound:

Lemma 5.3.4 $\max_{1\le i\le s} R_i \ge 1 - e^{-(1-1/k)}\alpha$.

Proof: To prove this lemma, we need to eliminate the explicit dependency of $\max_{1\le i\le s} R_i$ to the components of M, H, L and $\tilde{L}$.

First of all, for each t_i, we can replace $\tilde{L}(t_i)$ by $1 - \alpha - L(t_i)$ and simplify P_i as:

$$P_i = (\frac{k-1}{k}t_i + \frac{1}{k})(1-\alpha) - (\frac{k-1}{k}t_i + \frac{1}{k} - 1)L(t_i) + \sum_{j=1}^{i}\frac{M_j}{t_j} + \sum_{j=i+1}^{s}(\frac{k-1}{k}t_i + \frac{1}{k})\frac{M_j}{t_j}.$$

Similarly, we can replace $H(t_i)$ by $\alpha - M(t_i)$ and further replace $M(t_i)$ by $\sum_{j=1}^{i} M_j$ to get:

$$Q_i = \frac{k-1}{k}\alpha + \frac{1}{k}(1-\alpha) + \frac{k-1}{k}L(t_i) + \sum_{j=1}^{i}(\frac{1}{t_j} - \frac{k-1}{k})M_j + \sum_{j=i+1}^{s}\frac{1}{k}\frac{M_j}{t_j}.$$

Now we are ready to eliminate $L(t_i)$. Since $R_i = \max\{P_i, Q_i\}$, we have that

$$R_i \ge \frac{1}{t_i}P_i + \frac{t_i - 1}{t_i}Q_i = 1 - (\frac{k-1}{kt_i} + \frac{1}{k})\alpha + \sum_{j=1}^{s}\frac{M_j}{t_j} - \frac{k-1}{k}(1 - \frac{1}{t_i})\sum_{j=1}^{i} M_j.$$

At last, we will eliminate M_j for all j. Observe that $\max_{1\le i\le s} R_i$ is lower bounded by the average, we have the following:

$$\begin{aligned}
& s \max_{1\le i\le s} R_i \\
\ge\ & \sum_{i=1}^{s} R_i \\
\ge\ & s - \sum_{i=1}^{s}(\frac{k-1}{kt_i} + \frac{1}{k})\alpha + s\sum_{j=1}^{s}\frac{M_j}{t_j} - \sum_{i=1}^{s}\frac{k-1}{k}(1 - \frac{1}{t_i})\sum_{j=1}^{i} M_j \\
=\ & s - \sum_{i=1}^{s}(\frac{k-1}{kt_i} + \frac{1}{k})\alpha + s\sum_{j=1}^{s}\frac{M_j}{t_j} - \sum_{j=1}^{s}\big[\sum_{i=j}^{s}\frac{k-1}{k}(1 - \frac{1}{t_i})\big]M_j \\
=\ & s - \sum_{i=1}^{s}(\frac{k-1}{kt_i} + \frac{1}{k})\alpha + \sum_{j=1}^{s}\big[\frac{s}{t_j} - \sum_{i=j}^{s}\frac{k-1}{k}(1 - \frac{1}{t_i})\big]M_j.
\end{aligned}$$

Therefore, in order to eliminate M_j for all j, we only need to choose $t_1, ..., t_s$ such that

$$\frac{s}{t_j} - \sum_{i=j}^{s} \frac{k-1}{k}(1 - \frac{1}{t_i}) = 0 \tag{21}$$

for all $1 \leq j \leq s$.

As long as we can find such $t_1, ..., t_s$, we have:

$$\max_{1 \leq i \leq s} R_i \geq 1 - [\frac{k-1}{k} \cdot \frac{1}{s} \sum_{i=1}^{s} \frac{1}{t_i} + \frac{1}{k}]\alpha. \tag{22}$$

Observe that if we denote $\frac{1}{t_j}$ by z_j, (21) is equivalent to the system of linear equations $A \cdot \boldsymbol{z} = (s, s-1, ..., 1)^T$ where A is the matrix

$$\begin{pmatrix} \frac{k}{k-1}s+1 & 1 & ... & ... & 1 \\ 0 & \frac{k}{k-1}s+1 & 1 & ... & 1 \\ 0 & 0 & \frac{k}{k-1}s+1 & ... & ... \\ ... & ... & ... & ... & 1 \\ 0 & 0 & ... & 0 & \frac{k}{k-1}s+1 \end{pmatrix}$$

For the ease of notation, we use β to denote $\frac{k}{k-1}$. It is easy to see that A is invertible and its inverse is:

$$\begin{pmatrix} \frac{1}{\beta s+1} & -\frac{1}{(\beta s+1)^2} & -\frac{\beta s}{(\beta s+1)^3} & ... & -\frac{(\beta s)^{s-2}}{(\beta s+1)^s} \\ 0 & \frac{1}{\beta s+1} & -\frac{1}{(\beta s+1)^2} & ... & -\frac{(\beta s)^{s-3}}{(\beta s+1)^{s-1}} \\ 0 & 0 & \frac{1}{\beta s+1} & ... & ... \\ ... & ... & ... & ... & -\frac{1}{(\beta s+1)^2} \\ 0 & 0 & ... & 0 & \frac{1}{\beta s+1} \end{pmatrix}$$

Therefore, the system of equations has a unique solution $\boldsymbol{z} = A^{-1}(s, s-1, ..., 1)^T$. Moreover, we have the following:

$$
\begin{aligned}
& z_1 + ... + z_s \\
= \; & (1, 1, ..., 1)\big[A^{-1}(s, s-1, ..., 1)^T\big] \\
= \; & \big[(1, 1, ..., 1)A^{-1}\big](s, s-1, ..., 1)^T \\
= \; & \sum_{i=1}^{s} \frac{(\beta s)^{i-1}}{(\beta s + 1)^i}(s - i + 1) \\
= \; & \beta s\big(\frac{\beta s}{1 + \beta s}\big)^s + (1 - \beta)s.
\end{aligned}
$$

Therefore, if we replace z_i back by $\frac{1}{t_i}$ for all i, we have:

$$\frac{1}{s}\sum_{i=1}^{s}\frac{1}{t_i} = \beta\big(\frac{\beta s}{1 + \beta s}\big)^s + (1 - \beta)$$

and hence

$$\lim_{s\to\infty}\frac{1}{s}\sum_{i=1}^{s}\frac{1}{t_i} = \beta e^{-1/\beta} + 1 - \beta = \frac{k}{k-1}e^{-(1-1/k)} - \frac{1}{k-1}.$$

Together with (22), we have $\max_{1\le i\le s} R_i \ge 1 - e^{-(1-1/k)}\alpha$. □

Finally, we are ready to prove:

Theorem 5.3.5 *The approximation ratio of k-lookahead mechanism is at least $\frac{e^{1-1/k}}{1+e^{1-1/k}}$.*

Proof: Let rev_k be the revenue of the k-lookahead mechanism. From Lemma 5.3.4, we know that $\mathsf{rev}_k \ge 1 - e^{-(1-1/k)}\alpha$. Together with Lemma 5.3.1, we have

$$\mathsf{rev}_k \ge \max\{\alpha, 1 - e^{-(1-1/k)}\alpha\}.$$

Simple calculation shows that for all positive value x,

$$\max\{\alpha, 1 - x\alpha\} \ge \frac{1}{1+x}.$$

Therefore, we have $\mathsf{rev}_k \ge \frac{e^{1-1/k}}{1+e^{1-1/k}}$. This completes our proof. □

5.4 Tightness of Analysis

In the last section, we showed that the approximation ratio of k-lookahead is $\frac{e^{1-1/k}}{1+e^{1-1/k}}$. In particular, the 2-lookahead mechanism, which is of special interest, has an approximation ratio of at least $\frac{\sqrt{e}}{1+\sqrt{e}}$. In this section, we design an example to show that our analysis for 2-lookahead is tight. To do so, we need some definitions.

Since 2-lookahead auction is a deterministic mechanism, it either allocates the item, or does not allocate to anyone. We will consider the *empty instances* that a 2-lookahead mechanism doesn't allocate the item. We use $\mathsf{empt}(D)$ to denote the *empty probability* that empty instances occur on a distribution D. In the following, we will use $E_2(D)$ to denote the 2-lookahead mechanism with the maximum empty probability.

In a setting where there are only three bidders, we say that a distribution D is *valid*, if the third bidder always has valuation $v_3 = 1$ and the valuations of the other two bidders are at least 1.

We first prove a property of valid distributions. Let $\mathsf{rev}_2(D)$ and $\mathsf{opt}(D)$ denote the revenue of the 2-lookahead and the optimal auction for a distribution D respectively.

Lemma 5.4.1 *Let D be a valid distribution on three bidders, then we have* $\mathsf{opt}(D) \geq \mathsf{rev}_2(D) + \mathsf{empt}(D)$.

Proof: Consider this auction $\mathcal{A}$: run the 2-lookahead auction and if it allocates the item to bidder i in $K = \{1, 2\}$ with payment p, we still allocate the item to i with payment p. Otherwise we allocate the item to bidder 3 with payment 1. A remark here is that if the bids are not consistent with a valid distribution, we do not allocate the item. It is easy to see that $\mathcal{A}$ is truthful, and its revenue is $\mathsf{rev}_2(D) + \mathsf{empt}(D)$. Therefore, $\mathsf{opt}(D) \geq \mathsf{rev}_2(D) + \mathsf{empt}(D)$. □

The above lemma provides a lower bound of opt. In the following, we will explicitly construct a valid distribution D such that $\mathsf{empt}(D) \geq \frac{\mathsf{rev}_2(D)}{\sqrt{e}}$, hence prove our desired

ratio.

In our example, there are three bidders and the third bidder's valuation is always 1. Now we construct the distribution D_2 for the first two bidders explicitly. We assume that there are m possible valuations $p_0, p_1, \cdots, p_m (m$ is an *odd number*$)$. Then we define $x_0 = 1$ and $x_i = (1+p)^i - (1+p)^{i-1} = p(1+p)^{i-1}$ for $1 \leq i \leq m$ where p is a parameter. We will set the value of p and choose $p_1, ..., p_m$ later. One can see that a property of our construction is $\sum_{0 \leq i \leq j} x_i = (1+p)^j$ for all $j \leq m$.

Now consider this following distribution matrix D_2 where $D_2(i,j)$ denotes the probability of $v_1 = p_i, v_2 = p_j$:

p_m	$x_m x_0$	0	$\dots$	$\dots$	0
p_{m-1}	$x_{m-1}x_0$	$(x_{m-1}+x_m)x_1$	0	$\dots$	0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
p_1	x_1x_0	x_1x_1	$\dots$	$x_1(x_{m-1}+x_m)$	0
p_0	x_0x_0	x_0x_1	$\dots$	x_0x_{m-1}	x_0x_m
	p_0	p_1	$\dots$	p_{m-1}	p_m

Namely:

$$D_2(i,j) = \begin{cases} 0 & i+j > m \\ x_i x_j & i+j < m \\ x_i(\sum_{j \leq k \leq m} x_k) & (i+j=m) \text{ and } (i<j) \\ (\sum_{i \leq k \leq m} x_k)x_j & (i+j=m) \text{ and } (i>j) \end{cases}$$

In fact, D_2 should be normalized to become a distribution. However, since we only care about the ratio between $\mathsf{empt}(D)$ and $\mathsf{rev}_2(D)$, we will simply use D_2 as the distribution without normalizing. From now on, we will simply use E_2, rev_2 and empt to denote $E_2(D), \mathsf{rev}_2(D)$ and $\mathsf{empt}(D)$.

Now we choose $p_0 = 1$ and $p_i = \frac{\sum_{0 \leq j \leq m} x_j}{\sum_{j \geq i} x_j}$ for all $1 \leq i \leq m$. Therefore, we have $p_0 \leq p_1 \leq \cdots \leq p_m$. Furthermore,we obtain the following characterization of the event that E_2 allocates the item:

Lemma 5.4.2 *Let p_i, p_j be the bid of bidder 1 and 2 respectively, then E_2 allocates the item if and only if $i + j = m$.*

Proof: First of all, by our choice, it is easy to verify the following:

Property 5.4.3 *If $i < m/2$, then:*

$$p_0\big(\sum_{0\le k\le m-i} D_2(i,k)\big) = \cdots = p_j\big(\sum_{j\le k\le m-i} D_2(i,k)\big) = \cdots = p_{m-i}D_2(i,m-i) = x_i\sum_l x_l.$$

Basically, this property can be interpreted as follows: fix $v_1 = p_i$, the expected revenue obtained by offering bidder 2 a threshold price p_j is a constant when $0 \le j \le m - i$. As a result, recall that by Theorem 5.2.1, the winner in a single item auction pays the threshold price, we have:

Corollary 5.4.4 *In E_2, $t_2(v_1) \ge p_{m-i}$ when $v_1 = p_i$ for all $i < m/2$. Similarly, $t_1(v_2) \ge p_{m-j}$ when $v_2 = p_j$ for all $j < m/2$.*

The proof of the corollary is straightforward: Suppose $v_1 = p_i$ for some $i < m/2$. If $t_2(v_1) < p_{m-i}$, then we can always increase the threshold price to p_{m-i} without decreasing the revenue. By doing this, we only increase the empty probability. This is a contradiction to our assumption that E_2 maximizes the empty probability.

Now we are ready to prove the lemma. If it is not true, suppose $i + j < m$ but E_2 allocates the item to either bidder 1 or 2. Consider the smallest sum of $i + j$ that satisfies the above. Without lose of generality, we may assume $i < m/2$. From Corollary 5.4.4, since $i + j < m$, we know that bidder 2 can not get the item. Therefore, bidder 1 gets the item when $v_1 = p_i$ and $v_2 = p_j$. At the same time, $j > m/2$ otherwise we can get a contradiction from Corollary 5.4.4. So bidder 1 still gets the item when $v_1 = p_{m-j} > p_i$ and $v_2 = p_j$.

Now we show that we can modify the allocation of E_2 when $v_2 = p_j$ to get more empty probability and the expected revenue of modified auction is not less than the

original one. Let E_2' be an auction as follows: (1) it performs exactly the same as E_2 when $v_2 \neq p_j$ and (2) when $v_2 = p_j$, E_2' allocates the item to bidder 2 only when $v_1 = p_{m-j}$ and otherwise allocates nothing.

Obviously, E_2' has a larger empty probability than E_2. To get a contradiction, we only need to prove that its expected revenue is at least as large as E_2. In other words, we want to show:

$$p_j D_K(m-j, j) \geq p_i \sum_{k=i}^{m-j} D_K(k, j) \tag{23}$$

By our construction, simple calculation shows that (23) is equivalent to the following

$$\begin{aligned} & p(1+p)^{m-j-1}\big((1+p)^{j-1} - \sum_{l=0}^{i-1} x_l\big) \\ \geq & p(1+p)^{j-1}\big((1+p)^{m-j-1} - \sum_{l=0}^{i-1} x_l\big), \end{aligned}$$

which always holds when $j > m/2$. This is a contradiction. □

By the above characterization, we can easily calculate rev_2 and empt. We will show that by choosing the parameter p appropriately, $\mathsf{rev}_2 \leq \sqrt{e} \cdot \mathsf{empt}$, which implies:

Theorem 5.4.5 *The approximation ratio of 2-lookahead auction is at most* $\frac{\sqrt{e}}{\sqrt{e}+1}$.

Proof: By Lemma 5.4.2 and our construction, we first estimate rev_2 as follows:

$$\mathsf{rev}_2 \leq \sum_{i,j:i+j=m} p_j D_2(i,j) = 2\Big(\sum_{0 \leq i < m/2} x_i\Big)\Big(\sum_{l=0}^{m} x_l\Big) = 2(1+p)^{3m/2}.$$

Now we compute the empty probability empt. Again, by Lemma 5.4.2, we have

$\mathsf{empt} = \sum_{i,j:i+j<m} x_i x_j$, which can be calculate as follows:

$$\begin{aligned}
& \sum_{i,j:i+j<m} x_i x_j \\
= & \sum_{i=0}^{m-1} \sum_{j=0}^{m-1-i} x_i x_j \\
= & \; p(1+p)^{m-2} + \sum_{i=0}^{m-2} \sum_{j=0}^{m-1-i} x_i x_j \\
= & \; p(1+p)^{m-2} + \sum_{i=0}^{m-2} x_i (1+p)^{m-1-i} \\
= & \; p(1+p)^{m-2} + (1+p)^{m-1} + \sum_{i=1}^{m-2} x_i (1+p)^{m-1-i} \\
= & \; p(1+p)^{m-2} + (1+p)^{m-1} + \sum_{i=1}^{m-2} p(1+p)^{m-2} \\
= & \; p(1+p)^{m-2} + (1+p)^{m-1} + (m-2)p(1+p)^{m-2}
\end{aligned}$$

Now we set $p = 1/m$ and let $m \to \infty$, we have $\mathsf{rev}_2 \leq 2e^{3/2}$ and $\mathsf{empt} \geq 2e$. Therefore, $\mathsf{rev}_2 \leq \sqrt{e} \cdot \mathsf{empt}$. Therefore, by our previous argument, the approximation ratio of 2-lookahead auction is at most $\frac{\sqrt{e}}{\sqrt{e}+1}$. □

5.5 Discussion

Perhaps the first question that every theoretical computer scientist would ask here is whether the analysis of the k-lookahead mechanism can be improved in general. A nature attempt for this question from the negative aspect is to generalize our tight instance for 2-lookahead in section 5.4 to the k-lookahead for general k. In particular, one might consider the following distribution $D_K(i_1, ..., i_k)$ for the set K of the highest k bidders:

1. $D_K(i_1, \cdots, i_k) = 0$: there exists $p, q \in [k]$ such that $p \neq q$ and $i_p + i_q > m$.

2. $D_K(i_1, \cdots, i_k) = \prod_{j \in K} x_{i_j}$: for all $p, q \in [k](p \neq q)$, we have $i_p + i_q < m$.

3. $D_K(i_1, \cdots, i_k) = \prod_{j \in K \setminus \{l\}} x_{i_j} \cdot \sum_{j=i_l}^{m} x_j$: $\max_{p,q \in [k](p \neq q)} \{i_p + i_q\} = m$. Without lose of generality, we assume $i_l = \max\{i_1, \cdots, i_k\}$.

Again, we assume that the highest bid outside K is $v_{k+1} = 1$. Similar to the analysis for 2-lookahead, we can prove that k-lookahead allocates to some bidder in K if and only if $i_1, i_2, \cdots, i_k$ is such that $\max_{p,q \in [k](p \neq q)} \{i_p + i_q\} = m$. However, simple calculation implies that the ratio between the empty probability and the revenue of the k-lookahead is at most $2/k$. This only implies a $\frac{k}{k+2}$ upper bound on the approximation ratio of the k-lookahead mechanism. Therefore, to obtain better upper bound, if possible, one might need new ideas and techniques.

From the positive aspect, one might improve the analysis via the following approach: Instead comparing the revenue of k-lookahead to t-fixed price and t-pivot auctions, we could compare to more delicate auctions such as a hybrid of t_1-fixed price and t_2-pivot auctions for distinct values of t_1, t_2. Obviously, these auctions would provide better revenue, however, the difficulty here is to show that the hybrid mechanism is truthful.

Another interesting open question is to further close the gap between the revenues of the optimal deterministically truthful and truthful-in-expectation mechanisms. Our analysis of 2-lookahead implies that the gap is at most a factor of $\frac{1+\sqrt{e}}{\sqrt{e}}$. As we mentioned, our analysis is tight, hence closing the gap further requires better truthful-in-expectation mechanisms which can be derandomized.

5.6 Summary

In this chapter, we study the optimal auction design problem in the fundamental setting of single-item setting. We make no assumption on the distribution of the bidders' valuations. In particular, the bidders might not be independent. We study this problem from a computer science aspect, i.e. we design efficient truthful mechanisms that

obtain the approximate optimality. We show that the approximation ratio of the k-lookahead mechanism is at least $\frac{e^{1-1/k}}{1+e^{1-1/k}}$ and the analysis is tight for 2-lookahead. An interesting open question is to either improve the analysis of k-lookahead for general $k \geq 3$ or design examples to show that our analysis is already tight.

CHAPTER VI

CONCLUSION

As we have seen, problems in multi-agent systems introduce new challenges in the theory of algorithm design. In these problems, economic considerations such as optimization and game theory become necessary in their solutions. At the same time, due to the complex dependency of multiple agents, usually approximation serves as a guiding principle in solving these problems. As a consequence, approximation algorithm design in multi-agent systems becomes one of the most important areas in the theory of computing. In this thesis, we contribute to this area in two directions. We first generalize the classical combinatorial optimization to the submodular setting which fits into the multi-agent systems; then we design truthful mechanisms with approximate optimality for some auction environments. In fact, our work is also closely related to other directions in the multi-agent systems such as equilibrium computation, and we believe that the algorithmic and game-theoretic insights gained will help better understand the real-world and theoretical problems.

CHAPTER VII

BIBLIOGRAPHIC NOTE

Most of the research that appears in this thesis was published in theoretical computer science conferences.

Chapter 2 and 3 are based on two papers "Approximability of Combinatorial Problems with Multi-agent Submodular Cost Functions" and "Optimal Approximation Algorithms for Multi-agent Combinatorial Problems with Discounted Price Functions" ([18, 20]) with Gagan Goel, Chinmay Karande and Pushkar Tripathi.

In Chapter 3, the simple black-box reduction is a modification of the one given in the paper "Single Parameter Combinatorial Auctions with Partially Public Valuations" ([19]) with Gagan Goel and Chinmay Karande. The main result which is a factor reserving reduction is based on the paper "Black-box Reductions in Mechanism Design"([26]) with Zhiyi Huang and Yuan Zhou. Chapter 4 is a working paper with Xue Chen and Pinyan Lu.

REFERENCES

[1] Alon, N., Moshkovitz, D., and Safra, S., "Algorithmic construction of sets for k-restrictions," *ACM Trans. Algorithms*, vol. 2, no. 2, pp. 153–177, 2006.

[2] Ausubel, L. M. and Milgrom, P., "The lovely but lonely vickrey auction," in *Combinatorial Auctions, chapter 1*, MIT Press, 2006.

[3] Babaioff, M., Lavi, R., and Pavlov, E., "Single-value combinatorial auctions and implementation in undominated strategies," in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, 2006.

[4] Balcan, M.-F., Blum, A., Hartline, J. D., and Mansour, Y., "Mechanism design via machine learning," in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[5] Bar-Yehuda, R. and Even, S., "A linear time approximation algorithm for the weighted vertex cover problem," *Journal of Algorithms*, vol. 2, pp. 198–203, 1981.

[6] Bei, X. and Huang, Z., "Bayesian incentive compatibility via fractional assignments," in *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[7] Bollobas, B., *Random Graphs*. Cambridge University Press, 2001.

[8] Briest, P., Krysta, P., and Vocking, B., "Approximation techniques for utilitarian mechanism design," in *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

[9] Calinescu, G., Chekuri, C., Pal, M., and Vondrak, J., "Maximizing a submodular set function subject to a matroid constraint," in *Proceedings of the 12th international conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2007.

[10] Clarke, E. H., "Multipart pricing of public goods," *Public Choice*, vol. 11, September 1971.

[11] Cremer, J. and McLean, R. P., "Optimal selling strategies under uncertainty for a discriminating monopolist when demands are interdependent," *Econometrica*, vol. 53, pp. 345–61, March 1985.

[12] Dinur, I. and Safra, S., "On the hardness of approximating minimum vertex cover," *Annals of Mathematics*, vol. 162, no. 1, pp. 439–486, 2005.

[13] Dobzinski, S., Fu, H., and Kleinberg, R., "Optimal auctions with correlated bidders are easy," in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, 2011.

[14] Dughmi, S. and Roughgarden, T., "Black-box randomized reductions in algorithmic mechanism design," in *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.

[15] Feige, U., Mirrokni, V. S., and Vondrak, J., "Maximizing non-monotone submodular functions," in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[16] Feige, U. and Vondrak, J., "Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[17] Fleischer, L., Fujishige, S., and Iwata, S., "A strongly polynomial-time algorithm for minimizing submodular functions," *Journal of the ACM*, vol. 48, pp. 761–777, 2001.

[18] Goel, G., Karande, C., Tripathi, P., and Wang, L., "Approximability of combinatorial problems with multi-agent submodular cost functions," in *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2009.

[19] Goel, G., Karande, C., and Wang, L., "Single parameter combinatorial auctions with partially public valuations," in *Proceedings of the 3rd International Symposium on Algorithmic Game Theory (SAGT)*, 2010.

[20] Goel, G., Tripathi, P., and Wang, L., "Optimal approximation algorithms for multi-agent combinatorial problems with discounted price functions," in *Proceedings of the 30th Foundations of Software Technology and Theoretical Computer Science(FSTTCS)*, 2009.

[21] Goemans, M. X., Harvey, N. J. A., Iwata, S., and Mirrokni, V., "Approximating submodular functions everywhere," in *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2009.

[22] Groves, T., "Incentives in teams," *Econometrica*, vol. 41, pp. 617–631, July 1973.

[23] Hartline, J., Kleinberg, R., and Malekian, A., "Bayesian incentive compatibility via matchings," in *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[24] Hartline, J. D. and Lucier, B., "Bayesian algorithmic mechanism design," in *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, 2010.

[25] Hayrapetyan, A., Swamy, C., and Tardos, E., "Network design for information networks," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2005.

[26] Huang, Z., Wang, L., and Zhou, Y., "Black-box reductions in mechanism design," in *Proceedings of the 14th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems(APPROX)*, 2011.

[27] Iwata, S., "A faster scaling algorithm for minimizing submodular functions," *SIAM J. Comput.*, vol. 32, no. 4, pp. 833–840, 2003.

[28] Iwata, S., "Submodular function minimization," *Math. Program.*, vol. 112, no. 1, pp. 45–64, 2008.

[29] Iwata, S. and Nagano, K., "Submodular function minimization under covering constraints," in *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2009.

[30] Iwata, S. and Orlin, J. B., "A simple combinatorial algorithm for submodular function minimization," in *Proceedings of the 19th Annual ACM -SIAM Symposium on Discrete Algorithms (SODA)*, 2009.

[31] Johnson, D. S., "Approximation algorithms for combinatorial problems," in *Proceedings of the 5th Annual ACM symposium on Theory of Computing (STOC)*, 1973.

[32] Khot, S., "On the power of unique 2-prover 1-round games," in *Proceedings of the 34th annual ACM symposium on Theory of computing (STOC)*, 2002.

[33] Khot, S., Lipton, R. J., Markakis, E., and Mehta, A., "Inapproximability results for combinatorial auctions with submodular utility functions," in *Proceedings of the 1st Workshop on Internet and Network Economics (WINE)*, 2005.

[34] Khot, S. and Regev, O., "Vertex cover might be hard to approximate to within 2-epsilon," *J. Computer System Science.*, vol. 74, no. 3, pp. 335–349, 2008.

[35] Klemperer, P., "Auction theory: A guide to the literature," microeconomics, EconWPA, Mar. 1999.

[36] Lavi, R. and Swamy, C., "Truthful and near-optimal mechanism design via linear programming," in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[37] Mirrokni, V. S., Schapira, M., and Vondrak, J., "Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions," in *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, 2008.

[38] Myerson, R., "Optimal auction design," *Mathematics of operations research*, vol. 6, pp. 58–73, Feb 1981.

[39] Nisan, N. and Ronen, A., "Algorithmic mechanism design," in *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, 1999.

[40] Orlin, J. B., "A faster strongly polynomial time algorithm for submodular function minimization," in *The 14th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2007.

[41] Papadimitiou, C. and Pierrakos, G., "On optimal single-item auctions," in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, 2011.

[42] Papadimitriou, C., Schapira, M., and Singer, Y., "On the hardness of being truthful," in *In 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

[43] Raz, R. and Safra, S., "A sub-constant error-probability low-degree test, and sub-constant error-probability pcp characterization of np," in *Proceedings of the 29th annual ACM symposium on Theory of computing (STOC)*, 1997.

[44] Ronen, A., "On approximating optimal auctions," in *In The 3rd ACM Conference on Electronic Commerce (EC)*, 2001.

[45] Ronen, A. and Saberi, A., "Optimal auctions are hard," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[46] Schrijver, A., "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *Journal of Combinatorial Theory, Series B*, vol. 80, pp. 346–355, 2000.

[47] Sharma, Y., Swamy, C., and Williamson, D. P., "Approximation algorithms for prize collecting forest problems with submodular penalty functions," in *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2007.

[48] Sviridenko, M., "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letter*, vol. 32, no. 1, pp. 41–43, 2004.

[49] Svitkina, Z. and Fleischer, L., "Submodular approximation: Sampling-based algorithms and lower bounds," in *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

[50] Svitkina, Z. and Tardos, E., "Facility location with hierarchical facility costs," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm (SODA)*, 2006.

[51] Vazirani, V., *Approximation Algorithms*. Springer-Verlag, 2003.

[52] Vickrey, W., "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[53] Vondrak, J., "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[54] Wolsey, L. A., "An analysis of the greedy algorithm for the submodular set covering problem," *Combinatorica*, vol. 2, no. 4, pp. 385–393, 1982.