Representations and Parameterizations of Combinatorial Auctions

by

David Ryan Loker

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Combinatorial auctions (CAs) are an important mechanism for allocating multiple items while allowing agents to specify preferences over bundles of items. In order to communicate these preferences, agents submit bids, which consist of one or more items and a value indicating the agent's preference for these items. The process of determining the allocation of items is known as the winner determination problem (WDP). WDP for CAs is known to be NP-complete in the general case.

We consider two distinct graph representations of a CA; the bid graph and the item graph. In a bid graph, vertices represent bids, and two vertices are adjacent if and only if the bids share items in common. In an item graph, each vertex represents a unique item, there is a vertex for each item, and any bid submitted by any agent must induce a connected subgraph of the item graph. We introduce a new definition of combinatorial auction equivalence by declaring two CAs equivalent if and only if their bid graphs are isomorphic.

Parameterized complexity theory can be used to further distinguish between NP-hard problems. In order to make use of parameterized complexity theory in the investigation of a problem, we aim to find one or more parameters that describe some aspect of the problem such that if we fix these parameters, then either the problem is still hard (fixed-parameter intractable), or the problem can be solved in polynomial time (fixed-parameter tractable).

We analyze WDP using bid graphs from within the formal scope of parameterized complexity theory. This approach has not previously been used to analyze WDP for CAs, although it has been used to solve set packing, which is related to WDP for CAs and is discussed in detail. We investigate a few parameterizations of WDP; some of the parameterizations are shown to be fixed-parameter intractable, while others are fixed-parameter tractable. We also analyze WDP when the graph class of a bid graph is restricted.

We also discuss relationships between item graphs and bid graphs. Although both graphs can represent the same problem, there is little previous work analyzing direct relationships between them. Our discussion on these relationships begins with a result by Conitzer *et al.* [7], which focuses on the item graph representation and its treewidth, a property of a graph that measures how close the graph is to a tree. From a result by Gavril, if an item graph has treewidth one, then the bid graph must be chordal [16]. To apply the other direction of Gavril's theorem, we use our new definition of CA equivalence. With this new definition, Gavril's result shows that if a bid graph of a CA is chordal, then we can construct an item graph that has treewidth one for some equivalent CA.

Acknowledgments

This thesis has been a long time in the making, and is the culmination of a lot of time and effort. There are many people who are either directly or indirectly responsible for my completion. First and foremost, I would like to thank Margareta, whose comments, suggestions and criticisms were instrumental in many aspects of my thesis. But even more importantly, her encouragement and support were beyond measure. Margareta, you will forever have my gratitude as I could not have done this without you.

My thanks goes out to all of my friends, whose support was unwavering and who helped distract me when I needed it. A special thanks goes out to Omid, who was always ready to help out a friend.

I would also like to thank Kate Larson, who helped guide my thesis to completion, and her suggestions, words of wisdom and encouragement were necessary parts of my success. Thank you Kate.

Finally, I would like to thank my family for their ongoing and unconditional support in everything I do. I would especially like to thank my brother Steve and his wife Erika for being there when I needed them the most; I am eternally grateful. Also, it goes without saying that I would like to thank both my parents. They have always believed in me, and without them, none of this would be possible.

Contents

List of Figures

Chapter 1

Introduction

An allocation mechanism deals with the problem of finding the best way to allocate items to one or more individuals [35, 32]. Auctions are examples of commonly used allocation mechanisms, and they are used throughout today's economy to allocate objects, resources, and services to agents. The process of determining the allocation of items is known as the winner determination problem (WDP) [35, 1, 32, 9, 8]. The efficiency of WDP greatly affects its real-world applicability, as waiting a long time to determine a winner is not realistic or desirable.

We will consider a type of auction where multiple items are to be sold. Typically, when there are many items to auction the seller will hold multiple, sequential single-item auctions or parallel single-item auctions [25, 36, 3, 38, 45]. Unfortunately, these types of allocation mechanisms do not allow items to be sold in bundles, where the agents have preferences over those bundles [36, 46]. For instance, an agent's value for a bundle may not always be the sum of the agent's value for each individual item in that bundle. In some

situations, obtaining two items together is worth more to an agent than obtaining any one of the items and therefore, we ideally want a mechanism that allows agents to group items together into bundles and communicate their preferences over these bundles; at a very high level, a combinatorial auction (CA) is an allocation mechanism for auctioning multiple items which allows us to model this behaviour [32, 9, 8].

Combinatorial auctions can be useful as allocation mechanisms in many practical scenarios. For example, they have been proposed for allocating parts of the electromagnetic spectrum for wireless communication services [37, 8]. A wireless operator may wish to obtain licenses for the same frequency in neighbouring cities for more than the sum of the individual licenses, since their customers value roaming between cities. A combinatorial auction allows the wireless operator to specify exactly what group of cities it wants for a given frequency. Other application domains where combinatorial auctions have been used include truckload transportation, London bus routes, and industrial procurement [37, 9, 8].

One of the challenges of running a combinatorial auction compared to a standard auction is determining who has won the auction. In the types of auctions most people are familiar with, where only one item is being sold, agents submit bids (which can be thought of as amounts the agents are willing to pay if they win the auction) to the seller (the person running the auction), and the agent with the highest bid *wins* the item (i.e. is given the item). In a combinatorial auction, determining the winners is more complex since the bids of the agents are on bundles of items, which may overlap. Intuitively, a solution to the winner determination problem for combinatorial auctions is a collection of pairwise disjoint bundles of items such that the sum of the largest bids on those bundles is maximized. The bundles must be pairwise disjoint because an item can be allocated at most once.

The winner determination problem for combinatorial auctions is a known NP-hard problem from its reduction from 3-set packing (a known NP-hard problem) [29, 44]. Further, it is also know that the maximum clique problem, and thus the maximum independent set problem, reduces to WDP for CAs [46]. Hastad showed that the maximum clique problem is NP-hard to approximate within a factor of $n^{1-\epsilon}$ [24], and Sandholm showed via a reduction from the maximum clique problem that WDP for CAs is also hard to approximate [46]. More precisely, no polynomial-time algorithm that solves the winner determination problem for combinatorial auctions can guarantee a solution that is within a factor of $n^{1-\epsilon}$ of the optimum for any $\epsilon > 0$, where n is the number of distinct bundles of items for which there is at least one bid [46]. There are special cases that can be approximated slightly better [26, 27, 23, 20, 5, 21, 22], which are reviewed by Sandholm [46]. However, while such approximations as well as other heuristics can be used, they often compromise other economic properties that we might want to maintain [46, 48].

Since it is often very important that we solve WDP for CAs optimally, our work focuses on optimal solutions. However, as we just stated, WDP for CAs is NP-hard and is also NP-hard to approximate [29, 44, 24, 46]. In an attempt to solve this problem efficiently, we apply techniques from parameterized complexity theory, an area of computational complexity theory recently developed by Downey and Fellows, which we describe at the end of Section 1.1 [11]. Parameterized complexity has not yet been applied directly to WDP for CAs, although some previous solutions to restricted CAs can be reinterpreted in this manner, which we note when appropriate. We note that parameterized complexity has been used to solve problems such as independent set and set packing. As previously stated, maximum independent set reduces to WDP for CAs, and we will see in Chapter 3 how

WDP for CAs reduces to maximum independent set. Similarly, it is known that a weighted version of set packing is equivalent to WDP for CAs [44, 46, 9, 18]. We discuss the previous work relating to the set packing problem in Section 2.5, as well as how the problem relates to WDP for CAs.

To better understand the winner determination problem for combinatorial auctions, we also investigate relationships between two different representations of CAs. In Section 1.1, these representations are defined as bid graphs and item graphs. Both representations are used in the literature to study WDP for CAs [48, 7, 18]. In studying the relationships between bid graphs and item graphs, we also consider a new definition of combinatorial auction equivalence, and use a theorem by Gavril to devise a new technique for constructing item graphs [16]. Section 1.3 provides more details.

In this chapter, we first introduce some concepts and definitions at a high level in Section 1.1. When necessary, we provide some details to make our definitions clear. In Section 1.2 we present previous results, some of which are not directly related to our contributions, but are related to combinatorial auctions, which is the focus of our research. Also in Section 1.2, we discuss our contributions that are related to previous results for solving WDP for CAs. In Section 1.3, we describe our contributions in some detail. Finally, in Section 1.4, we give a roadmap of the thesis.

1.1 Definitions

The problem we study involves auctions, which are examples of allocation mechanisms. In this section we present informal definitions and notation related to allocation mechanisms

and auctions. We also informally define the notions of a parameterized problem and fixed-parameter tractability because these are related to our results, which are outlined in Section 1.4. We refer the reader to Chapter 2 for a more formal treatment.

Allocation mechanisms have certain properties in common. We define *the system* of a mechanism to whatever is implementing or running the mechanism, or both. In an auction, the system is referred to as *the seller*. An allocation mechanism involves a set of agents $A = \{1, 2, \ldots, |A|\}$ and the following: a collection of non-empty strategy sets, one per agent, which define how agents can communicate with the system; an output function, which determines how to allocate items to the agents; and a collection of possible outcomes, where each outcome specifies an allocation of items to agents. The output function of a mechanism maps a collection of strategy sets, one per agent, to an outcome. Informally, one may think of the output function as mapping the communication, which is received by the system from agents, to an outcome.

The agents are responsible for communicating information to the system, should they wish to participate in the mechanism. This information specifies each agent's preference over each possible outcome of the mechanism. Each agent knows their true preference over each possible outcome, but does not necessarily communicate the truth to the system. That is, an agent may communicate preferences that differ from their true preferences. We leave the details to Chapter 2.

There are three properties that we require our mechanism to have, which we informally define to help understand why we will be interested in optimally computing a mechanism's output function. Each property is based in economics and can be defined mathematically, which we will see in Chapter 2. The properties are desirable from an economics perspective.

First, we require that our mechanism allow agents to report their preferences directly, without any extra work; we refer to this as *Property 1*. Second, we require that our mechanism provide incentives for agents to report their true preference over each possible outcome; we refer to this as *Property 2*. We require that our mechanism exhibit Property 2 in order to make guarantees about the quality of our mechanism's output. Specifically, if agents do not tell the truth, then we cannot guarantee that we have allocated the items in a way that the seller desired. Finally, we require that our mechanism is designed such that agents will want to participate; we refer to this as *Property 3*. There are principles described in Chapter 2 that guarantee these three properties, and there exist mechanisms that exhibit all three properties; for example, Vickrey-Clarke-Groves mechanisms [51, 6, 19, 32]. In order to ensure that a mechanism has all three properties, we typically restrict the way agents calculate how much they value each outcome. Fortunately, this restriction is simple and still allows plenty of flexibility [35, 32]. The details of how agents are restricted are not important to this chapter, and are left for discussion in Chapter 2.

It is not always easy to ensure that our mechanism has Properties 2 and 3. In this section, we will see how our ability to calculate a mechanism's output function optimally can affect whether or not our mechanism exhibits these properties. This is important because in Chapters 3 and 4, we search for ways to optimally calculate a mechanism's output function. First, we briefly introduce auctions as examples of allocation mechanisms so that we can define combinatorial auctions and construct two graphical representations of auctions, which are necessary for understanding our results.

As previously stated, an auction is an example of an allocation mechanism in which items are allocated to one or more agents, and the process of determining the allocation of

the items to agents is known as the winner determination problem. The majority of our discussion and all of our results in Chapters 3 and 4 pertain to a special type of auction known as a combinatorial auction, and solving the winner determination problem for such an auction. In auctions, the agents are often referred to as *bidders*, but for consistency we will continue to refer to them as agents. We denote the set of items by I, and we refer to a subset $S \subseteq I$ as a *bundle of items*. If an agent submits a bid on a bundle of items $S \subseteq I$, $|S| > 1$, then the agent is said to have a *preference* over the bundle of items.

Standard methods for auctioning multiple items include sequential single-item auctions, parallel single-item auctions and combinatorial auctions, which we discuss in more detail in Chapter 2. The first two methods approximate auctions involving multiple items through the use of multiple single-item auctions, and they are well studied in the literature [25, 36, 3, 38, 45]. In these two approaches, agents cannot communicate preferences over bundles of items. Using either approach, it is not possible to guarantee that all agents receive only the items they want, unless no agent has a preference over any bundle of items with more than one item [36, 46]. Under these auction methods, we cannot guarantee that our auction has Property 2 because the optimal solution to WDP may not be found [35, 32].

The last approach, which is the combinatorial auction approach, is a generic technique for auctioning multiple items. Informally, a *combinatorial auction* is an allocation mechanism that consists of a set of agents A and a set of items I to be auctioned off simultaneously. Each agent is assumed to know how much they value each bundle of items, which is equivalent to knowing their preferences. Because we are dealing with monetary denominations when auctioning items, the value of each bundle is a non-negative integer. We denote agent i's bid by B_i, and $B_i(S)$ denotes agent i's value of bundle of items S.

Combinatorial auctions allow agents to express preferences over bundles of items [32, 8], and are the focus of our work. In order to guarantee that our combinatorial auction has Properties 2 and 3, we almost always need an optimal solution to WDP. There are a few restrictions on combinatorial auctions for which approximate algorithms allow the auction to still exhibit Properties 2 and 3 [34, 14, 39, 2], but we focus on conditions where this is not the case. Since an optimal solution to WDP for CAs is often needed, there are multiple optimal algorithms and techniques that have been investigated [44, 47, 46, 48, 18]. We refer the reader to Section 2.3 for more details on combinatorial auctions.

Now that we have reviewed some of the notation of combinatorial auctions, we informally define the winner determination problem for combinatorial auctions. We give a more formal treatment in Section 2.3.1. If agent i submits a bid on a bundle of items S, then $B_i(S) > 0$. If agent i wishes to bid 0 on a bundle of items S, then they do so by not submitting a bid on S. We restrict our attention to only the highest bid for each bundle of items. Thus, for all $S \subseteq I$ such that at least one agent submitted a bid on S, we define $B^*(S)$ as $B^*(S) = \max\{B_i(S) \mid B_i(S) \text{ submitted}\}$. A solution to WDP for CAs is then a collection of pairwise disjoint bundles of items such that the sum of the bids defined by B^* is maximized. A bundle of items is allocated to an agent that submitted the largest bid for that bundle, with ties broken arbitrarily. If there are items not allocated by the solution, then these items are kept by the seller. We let the collection of distinct bundles of items S for which there is at least one bid on S be denoted by $\mathcal{S}^*$.

To analyze the winner determination problem for combinatorial auctions, we consider two known representations of CAs. The first representation we consider is known as a bid graph, which we now describe informally. For each $S \in \mathcal{S}^*$, we have exactly one vertex in

the bid graph. Each vertex in the bid graph then represents a bundle of items $S \in \mathcal{S}^*$. Further, each vertex is associated with the bid $B^*(S)$ and one agent, i say, that submitted bid $B_i(S) = B^*(S)$, with ties broken arbitrarily. Two vertices are adjacent in the bid graph if and only if their bundles of items are not disjoint. For each vertex v, we label v by $\{i, S_v, B^*(S_v)\}$, where $S_v \in \mathcal{S}^*$ is the bundle of items that vertex v represents, and i is the agent that submitted a bid $B_i(S_v) = B^*(S_v)$, with ties broken arbitrarily. See Section 2.3.3 for more details and Figures 2.4 and 2.6 for examples of bid graphs.

The bid graph of a combinatorial auction can be constructed in time polynomial in the size of the auction, and there is exactly one bid graph representation for any given combinatorial auction [46, 48]. The solution to WDP for CAs is equivalent to finding a maximum weighted independent set of the auction's bid graph, where the weight of each vertex is its associated $B^*(S)$ value [46, 48].

The second representation we consider is referred to as an item graph. To define an item graph, we briefly define three graph-theoretic notions. A *path* is an open walk in which no vertex or edge repeats itself. A graph is said to be *connected* if there exists a path between every pair of distinct vertices in the graph. An *induced subgraph* of a graph $G = (V, E)$ on a set of vertices $V' \subseteq V$ is the set of vertices V' together with every edge whose endpoints are both in V'. Informally, a *valid item graph* of a given combinatorial auction is a graph $G = (V, E)$, where V is the set of items. Further, for each $S \in \mathcal{S}^*$, the induced subgraph of the item graph on S must be connected. See Section 2.3.4 for more detail and Figure 2.5 for examples of item graphs.

In Chapter 3, we analyze the NP-hard winner determination problem on combinatorial auctions using parameterized complexity. We wish to find efficient algorithms for solving

WDP. The study of parameterized complexity theory, which was introduced by Downey and Fellows, is motivated by a desire to find efficient solutions to NP-hard problems and to classify NP-hard problems into various categories [11]. Intuitively, a problem in which we fix a parameter is called a *parameterized problem*. For example, one might fix the size of the maximum clique in a graph for a graph theoretical problem, or ask that the integer solution to a problem be bigger than some fixed integer k; both examples result in parameterized problems.

We make use of parameterized complexity to distinguish between intractable problems so that we can find parameterized problems that are (relatively) less computationally hard. As we will see in Chapter 3, restrictions can be made to combinatorial auctions that result in parameterized problem definitions for the winner determination problem that are either fixed-parameter tractable (easy), or fixed-parameter intractable (hard); we define both concepts in full detail in Section 2.4. We omit the details of parameterized complexity theory from this chapter, and simply refer to problems as either fixed-parameter tractable or fixed-parameter intractable; the latter is relatively harder than the former.

1.2 Previous Work and Related Contributions

There are a few existing algorithms that optimally solve WDP for general CAs. In this section, we first review four such algorithms, which do not involve bid graphs or item graphs. With the exception of the dynamic programming algorithm presented by Rothkopf *et al.*, these algorithms are not directly related to our contributions in Chapters 3 and 4, and so will not be discussed in any detail beyond that given in this section [44].

The weighted set packing problem is exactly the same problem as WDP for CAs [44, 46, 9, 18]. Any algorithm solving weighted set packing can be applied directly to WDP for CAs. We give an outline of previous solutions to the weighted set packing problem and how they can be used to solve WDP for CAs in Chapter 2, Section 2.5.

At the end of this section, in addition to the algorithms that optimally solve WDP for general CAs, we review solutions to WDP for restricted CAs. Previous algorithms that are related to our contributions and solve WDP in polynomial time were introduced by Rothkopf *et al.*, Sandholm, Sandholm and Suri, and Conitzer *et al.* [44, 46, 48, 7]. These algorithms are presented in Chapter 4, where appropriate. We also briefly and informally present these results at the end of this section, along with how our contributions relate to these results. We also review algorithms that are not directly related to our results, but are related to combinatorial auctions.

We first review an algorithm that was introduced by Sandholm *et al.* [47] that optimally solves WDP for general CAs. The algorithm iterates over all possible collections of pairwise disjoint bundles of items, and finds the collection that maximizes the sum of the bids defined by B^*. The number of possible collections of pairwise disjoint bundles of items is $O(|I|^{|I|})$ and $\omega(|I|^{|I|/2})$ [47]. Since the number of possible collections of pairwise disjoint bundles of items increases rapidly as the number of items $|I|$ increases [47], this is a very computationally expensive method for optimally solving WDP for CAs.

A dynamic programming algorithm, presented by Rothkopf *et al.*, has a running time of $o(2^{2|I|})$ and $\Omega(2^{|I|})$. The algorithm works by considering $B^*(S)$ for every $S \subseteq I$ in order to determine a solution to WDP. The maximal bid value for each set S is either $B^*(S)$, or it is the sum of the maximal bid values of two disjoint subsets S_1, S_2 of S, where $S_1 \cup S_2 = S$.

Rothkopf *et al.* proceed to calculate the maximal bid values by beginning with the smallest possible S and increasing its size until $S = I$. In Chapter 3, we see this algorithm in more detail and how it translates to a parameterized problem that is fixed-parameter tractable.

The next algorithm we review was presented by Sandholm [46]. The algorithm, while still exponential in the number of items, performs better than the dynamic programming algorithm by Rothkopf *et al.* as the number of submitted bids decreases. The algorithm allows bidding to occur on all combinations of items, avoids redundancy, capitalizes heavily on most bundles of items being valued as zero, and solves WDP for CAs optimally. The required running time for the algorithm, $O(|I| \cdot (|\mathcal{S}^*|/|I|)^{|I|})$, is expressed in terms of the number of items, $|I|$, and the number of distinct bundles of items S for which there is at least one bid on S, $|\mathcal{S}^*|$. The details of Sandholm's algorithm are not presented here because they are beyond the scope of this thesis. We refer the reader to Sandholm for more details [46].

We now present an algorithm that was introduced by Sandholm and Suri, which is a binary search type algorithm [48]. Its running time is slightly better than the algorithm above by Sandholm [46]. Each node in the binary search tree represents a bid $B^*(S)$, where $S \in \mathcal{S}^*$. For each node, we either include its bid in a solution to WDP, or we do not include its bid. We consider any node N and denote the bid it represents by $B^*(S)$. The subtree rooted at the left child of N involves only solutions to WDP that contain S. Therefore, if we traverse the tree by moving to N's left child during our search, then we add S to our current solution of WDP, and remove all bids from $\mathcal{S}^*$ that share items with S. We then recursively solve WDP by continuing our search in the subtree rooted at the left child of N. Conversely, the subtree rooted at the right child of N involves only

solutions to WDP that do not contain S. Therefore, if we traverse the tree by moving to N's right child during our search, then we remove S from $\mathcal{S}^*$. We then recursively solve WDP by continuing our search in the subtree rooted at the right child of N. The leaves of the tree represent possible solutions to WDP; we must find the solution with the maximum summation of bid values by searching the whole tree. We let k be the number of items in a bid with the smallest number of items. The algorithm we just described by Sandholm and Suri has a running time of $O((\frac{|\mathcal{S}^*|}{\lfloor |I|/k \rfloor} + 1)^{\lfloor |I|/k \rfloor})$ [48].

In addition to optimal solutions to WDP for general CAs, there are algorithms that solve WDP for restricted versions of CAs. Rothkopf *et al.* present a number of restrictions to the bids of a CA that allow us to find optimal solutions to WDP in polynomial time [44]. If the bids have at most two items each, WDP can be solved optimally in $O(|I|^3)$ time using an algorithm for maximum weighted matching [44]. WDP becomes NP-complete if the bid can have three items, which can be proven via a reduction from 3-set packing [44]. If the bids are of length 1, 2, or greater than $|I|/c$, then WDP can be solved optimally in $O(c \cdot (n_{long})^{c-1} \cdot |I|^3)$ time, where n_{long} is the number of bids of length greater than $|I|/c$ [44]. Rothkopf *et al.* also consider laying items out as leaf nodes of a tree and allowing agents to bid on any node (internal or leaf). A bid of value p submitted on a node N is a bid on the bundle of items S, where S contains all of the items that occur in the subtree rooted at N. WDP for such a CA can be solved optimally in $O(|I|^2)$ time [44]. These results by Rothkopf *et al.* are not directly related to the contributions of this thesis.

We now present previous solutions to WDP for restricted CAs that are related to our contributions in Chapter 4. We begin by briefly reviewing results presented by Rothkopf *et al.* and Sandholm and Suri [44, 48]. Rothkopf *et al.* consider ordering items and requiring

that bids only be placed on consecutive items [44]. Using dynamic programming, Rothkopf *et al.* show that WDP can be solved optimally for such an auction in time at most $O(|I|^2)$ [44]. In this instance, the bid graph is by definition an interval graph, which we formally define in Chapter 2, where the number of vertices in the graph is $|\mathcal{S}^*|$. The maximum weighted independent set of an interval graph can be solved in linear time in the number of vertices [13]. However, this method does not take into account the time required to construct the bid graph, which is discussed in Chapter 3 and is much larger than $O(|\mathcal{S}^*|)$. Sandholm and Suri present an algorithm for solving WDP for such an auction that runs in time $O(|I| + |\mathcal{S}^*|)$ [48]. In Chapter 4, we see how these results relate to item graphs and bid graphs. We note that a bid is said to *wrap around* an ordered list of items if it consists of two bids B_1 and B_2, where B_1 and B_2 are bids on consecutive items in the list, B_1 contains the first item in the list, and B_2 contains the last item in the list. Rothkopf *et al.* observe that if instead of only being able to place bids on consecutive items, we also allow bids to wrap around, then WDP can be solved optimally in time $O(|I|^3)$ by running the $O(|I|^2|)$ algorithm $|I|$ times [44]. Using Sandholm and Suri's faster algorithm for the previous restriction, we can solve WDP for CAs that allow wrap-around in time at most $O(|I| \cdot (|I| + |\mathcal{S}^*|))$ [48]. In this case, the bid graph would be a circular arc graph, which we define in Chapter 2. The maximum weighted independent set of a circular arc graph can be solved in time $O(l|\mathcal{S}^*|)$, where l is the minimum number of arcs passing through some point on the circle [49].

Finally, Conitzer *et al.* present a polynomial-time solution to WDP for restricted CAs, which relates directly to the item graph representations of the CAs [7]. First, we intuitively define the treewidth of a graph to be a measure of how close the graph is to

a tree. If a graph has a treewidth of one, then that graph is a tree. As the treewidth of a graph increases, the graph becomes less like a tree. Conitzer *et al.* show that if there exists a valid item graph G_I for a CA that has treewidth at most tw, and we are given a special structure T representing G_I, then the optimal solution to WDP for the CA can be determined in time $O(|T|^2(|\mathcal{S}^*|+1)^{tw+1})$ using dynamic programming [7]. In Chapter 4, we discuss a parameterized problem directly related to this result. Further, we investigate the relationship between valid item graphs of treewidth at most tw and bid graphs. In addition, we address concerns related to the computational hardness of constructing valid item graphs of treewidth at most tw, which were presented by Gottlob and Greco [18]. We address these concerns by considering a new definition of combinatorial auction equivalence, which intuitively involves isomorphic bid graphs, and present a construction technique using a theorem by Gavril [16]. The technique constructs item graphs of smaller treewidth from those of larger treewidth, if the bid graph has a particular structure.

1.3 Contributions

Our contributions can be divided into two categories: those resulting from parameterizations of WDP for CAs that involve bid graphs, and those that result from relationships between item graphs and bid graphs. In the former category, our contributions make extensive use of the bid graph representation of combinatorial auctions. Using parameterized complexity theory as a new approach to analyzing the complexity of structured versions of WDP, we formulate parameterized versions of the winner determination problem by parameterizing aspects of combinatorial auctions that pertain to their bid graphs. For

example, we consider bounding $|\mathcal{S}^*|$, the number of distinct bundles of items S such that there is at least one bid on S, from above by a parameter. We show that this parameterized problem instance is fixed-parameter tractable. Further, we consider restricting the structure of the bid graphs of combinatorial auctions, and then construct parameterized versions of WDP in this context. We then analyze the parameterized hardness of our various parameterized problem instances, and prove technical lemmas and theorems. Some of our parameterized versions of WDP are fixed-parameter tractable, while others are fixed-parameter intractable. We also discuss a parameterization of WDP that relates to a previous algorithm by Rothkopf *et al* [44].

One of the main contributions of this thesis is its demonstration of the use of parameterized complexity in the investigation of WDP for CAs. With parameterized complexity theory, it is possible to parameterize WDP and gain insight into new efficient algorithms, as well as prove when parameterizations of WDP are hard. We hope that this methodology will open up questions and ideas that lead to improved ways of solving and structuring WDP for CAs.

In the latter category, our contributions come from relationships between item graphs and bid graphs. More precisely, we consider item graph representations of a combinatorial auction and how they relate to the auction's bid graph. Despite the fact that item graphs and bid graphs differ in how they represent combinatorial auctions, the auctions they portray are the same. We investigate classes of item graphs and whether they enforce enough structure on their combinatorial auctions to define the class of bid graphs associated with these auctions. Further, we look beyond a direct relationship between item graphs and bid graphs and discuss a new definition of bid graph (and combinatorial auction)

equivalence, which allows us to present a construction technique for item graphs that is derived from a proof by Gavril [16]. This construction technique involves modifying the underlying combinatorial auction while maintaining essentially the same bid graph, in order to find a simpler item graph representation of the auction.

One of the most important contributions of this thesis comes from Chapter 4, where we introduce a new definition of combinatorial auction equivalence, which allows us to present a technique for constructing item graphs that has not been previously discussed. A result by Gottlob and Greco [18], which shows that it is NP-hard to decide whether or not a combinatorial auction has a valid item graph of treewidth three, brings to question the practicality of studying item graphs of bounded treewidth. This new definition of CA equivalence, as well as the construction technique derived from a result by Gavril [16], opens another avenue of investigation for determining how useful item graphs of bounded treewidth are in practice.

1.4 Thesis Outline

The remainder of this thesis is organized into four chapters. In Chapter 2, we present all of the notation and definitions that will be used throughout the remainder of the thesis.

In Chapter 3, we analyze combinatorial auctions via their bid graph representations. We make use of parameterized complexity theory in our analysis by formulating parameterized versions of the winner determination problem for combinatorial auctions. We also consider restricting the structure of the bid graphs of combinatorial auctions, and then use parameterized complexity theory to analyze parameterized versions of WDP in this

context.

In Chapter 4, we discuss the relationship between item graphs and bid graphs. We investigate structured item graphs and whether we can define the structure of bid graphs associated with combinatorial auctions for which these item graphs are valid. Further, we discuss a new definition of combinatorial auction equivalence, and present a construction technique for item graphs based on a theorem by Gavril [16]. Using this new definition of equivalence, we re-evaluate the relationship between item graphs and bid graphs.

Finally, in Chapter 5, we review our conclusions and present avenues for future research.

Chapter 2

Preliminaries

The concepts and notation we introduce in this chapter touch on a few different subjects, including graph theory, mechanism design, auctions, and parameterized complexity. Each section of this chapter introduces the ideas necessary for us to investigate the computational complexity of various formulations of a problem on combinatorial auctions. First, in Section 2.1 we cover key concepts in graph theory. Graphs will be used to represent combinatorial auctions, and are a starting point for the majority of our analysis. Further, in order to understand the underlying principles behind a problem we will investigate in detail, we provide an introduction to mechanism design in Section 2.2. Auctions are a specific application of allocation mechanisms from Section 2.2, and auction theory will be a main component in our discussion. Section 2.3 details all of the technical notions necessary to understand the auctions of interest. Our investigation will involve the marriage of both auction theory and complexity theory, but more specifically auction theory and parameterized complexity. We will use parameterized complexity to understand how algo-

rithms for problems on auctions compare to one another. Section 2.4 describes the notion of parameterized complexity and how it might be used. Finally, in Section 2.5, we detail the set packing problem. Specifically, we define the set packing problem, review previous work done on the problem, and describe how the problem relates to combinatorial auctions and their solutions.

We assume a basic knowledge of computational complexity theory; for further discussion and details, see Garey and Johnson [15]. We refer the reader to "Graph Theory" by Diestel [10] for a more in-depth treatment of graph theoretic notions. The introduction on mechanism design and auction theory supplied in Section 2.2 and Section 2.3 is by no means a complete specification of either area. We direct the reader to Krishna [32] and Mas-Colell *et al.* [35] for a more in-depth review of auctions and mechanisms. The more specific topic of combinatorial auctions is also covered in Section 2.3; however, not all details are covered. We direct the reader to "Combinatorial Auctions," by Cramton *et al.* for a very thorough and comprehensive review of combinatorial auctions [8]. For further details on parameterized complexity theory beyond that provided in Section 2.4, see Downey and Fellows [11].

2.1 Graph Theory

We will show how to use a graph to represent how bidders bid in an auction, and then use the structure of the constructed graph to determine the computational hardness of computing the winner(s) of the auction. In order to understand this construction and its structure, we introduce graphs as well as some specific classes of graphs. We will insist

that our constructed graph belongs to one of these specific graph classes and analyze each class' effect on the computational complexity of computing the winner(s) of the auction.

We denote a *graph* on vertices V with edges E as $G = (V, E)$. We may denote a vertex $v \in V$ of G as $v \in G$ when the meaning is clear. We assume that G is an *undirected graph* (i.e. edges are unordered pairs from $V \times V$).

An edge is *multiple* if there is another edge with the same endpoints, otherwise it is *simple*. A *loop* is an edge in which both endpoints are the same vertex. A graph is *simple* if it contains no loops or multiple edges. We assume that all of our graphs are simple.

A *walk* is an alternating sequence of vertices and edges, both beginning and ending with a vertex, in which each each edge has as endpoints the vertices immediately preceding and following it. A walk is *closed* if its first and last vertices are the same, otherwise it is *open*.

A *path* is an open walk in which no vertex or edge repeats itself. A *cycle* is a closed walk in which no vertex or edge repeats itself, with the exception that the first vertex must be the same as the last. The *length* of a path or cycle is the number of edges in its walk. In terms of notation, we ignore the edges in the walks of both paths and cycles. We denote a path as a list of vertices $i, i_1, i_2, \ldots, i_k, j$ such that there is an edge between any two consecutive vertices in the list. Similarly, we denote a cycle as a list of vertices such that there is an edge between any two consecutive vertices in the list.

An induced subgraph of a graph $G = (V, E)$ on a set of vertices $V' \subseteq V$ is the set of vertices V' together with any and every edge whose endpoints are both in V'.

Using the above definitions, we can further classify graphs. A graph is said to be *connected* if there exists a path between every pair of distinct vertices in V. An *acyclic* graph is a graph that has no cycles. We can also look at enforcing a more complicated

structure within a graph. For instance, a graph is called *bipartite* if its set of vertices can be partitioned into two disjoint sets, $V = \{V_1, V_2\}$, where no two vertices in V_i are adjacent. A graph *class* is a subset of all graphs that conform to some specified structure. For instance, the *bipartite graph class* consists of all graphs that are bipartite.

A *connected component* of a graph $G = (V, E)$ is the induced subgraph of a set of vertices V', such that the following holds: $V' \subseteq V$; the induced subgraph of G on V' is connected; and there does not exist any other V'' with $V' \subseteq V'' \subseteq V$ such that the induced subgraph of G on V'' is connected. When we refer to a component of a graph, we are referring to a connected component.

There are numerous ways of restricting the structure of a graph, which lead to countless possible graph classes. In addition to those described above, we will be interested in a few restrictions that lead to the following graph classes: trees, forests, interval graphs, chordal graphs, and graphs of bounded treewidth. A *tree* is a connected acyclic graph. A *forest* is a graph in which each connected component is a tree. To define interval graphs, circular arc graphs and chordal graphs, we make use of a few more definitions.

Let $S_1, S_2, \ldots, S_n$ be a family of sets and define the *intersection graph* $G = (V, E)$ of $S_1, \ldots, S_n$ as follows:

$$V = \{S_1, S_2, \ldots, S_n\}, \text{ and}$$
$$\{S_i, S_j\} \in E \iff S_i \cap S_j \neq \emptyset.$$

An *interval* is a connected portion of the real line. For instance, the interval $[1, 5]$ contains all real values between 1 and 5, including both end points. An *open interval* is

an interval that does not contain one or both of its end points. For example, the open interval $(3, 7]$ contains all real values between 3 and 7, including 7 but not including 3. Let $I_1, I_2, \ldots, I_n$ be intervals on the real line. Then, we define the *interval graph* $G = (V, E)$ representing the intervals $I_1, \ldots, I_n$ as follows:

$$V = \{I_1, I_2, \ldots, I_n\}, \text{ and}$$
$$\{I_i, I_j\} \in E \iff I_i \cap I_j \neq \emptyset.$$

As an example, we define $I_1 = (2, 4), I_2 = [3, 5], I_3 = (4, 9], I_4 = [5, 7], I_5 = [8, 10)$, and $I_6 = [1, 10]$. See Figure 2.1(a) for a depiction of intervals $I_1, \ldots, I_6$. Intervals I_1 and I_3 do not intersect because neither includes 4. Intervals I_2 and I_4 intersect because they both include 5. Figure 2.1(b) shows the interval graph of intervals $I_1, \ldots, I_6$. Consider the vertex corresponding to interval I_6. Vertex I_6 is adjacent to all other vertices because all intervals are subintervals of interval I_6. As another example, consider vertex I_1 corresponding to the interval I_1. This vertex is adjacent only to vertices I_6 and I_2 because interval I_1 only intersects intervals I_6 and I_2.

A circular arc graph is an extension of an interval graph. An *arc* of a circle is a continuous segment of the circumference of the circle. A *sub-arc* of an arc is a continuous segment of the arc. Two arcs Arc_1 and Arc_2 of a circle *intersect* if there exists an arc of the circle that is also a sub-arc of Arc_1 and Arc_2. Let $I_1, I_2, \ldots, I_n$ be arcs of a circle. Then we define the *circular arc graph* $G = (V, E)$ representing the arcs $I_1, \ldots, I_n$ as follows:

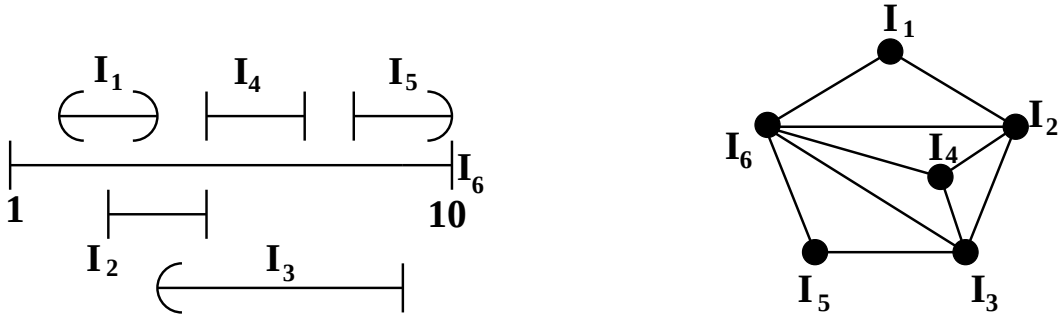


Figure 2.1: An example of an interval graph.

$$V = \{I_1, I_2, \ldots, I_n\}, \text{ and}$$

$$\{I_i, I_j\} \in E \iff \text{arc } I_i \text{ intersects arc } I_j$$

As an example, see Figure 2.2(a) for the definition of arcs $I_1, \ldots, I_6$. The arcs are drawn away from the circumference of the circle itself in order to better illustrate where their segments begin and end. The circular arc graph of the arcs $I_1, \ldots, I_6$ is shown in Figure 2.2(b). To explain the example in more detail, we consider vertex I_2 corresponding to arc I_2. Arc I_2 intersects arcs I_1, I_3 and I_4 on the circle, and thus in the circular arc graph vertex I_2 is adjacent to vertices I_1, I_3 and I_4.

Next, we define the class of chordal graphs. A *chord* is an edge between two vertices of a cycle, and the edge must not be a part of the cycle itself. See Figure 2.3 for example graphs where the dotted edges are chords. A *chordal graph* is a graph in which each cycle of length four or more has at least one *chord*. In contrast, a graph is not a chordal graph

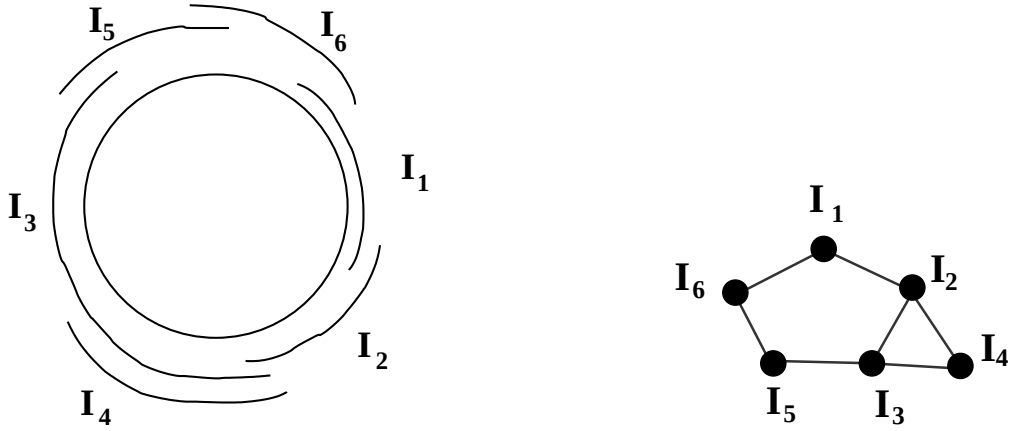


Figure 2.2: An example of a circular arc graph.

if it contains a cycle of length at least four with no chords. As shown in Figure 2.3(a), a cycle of length four with one chord is a chordal graph. The cycle of length five with four chords depicted in Figure 2.3(b) is also a chordal graph. See Figure 2.3(c) for an example of a graph that is not chordal. The cycle $1, 2, 4, 5, 1$ in Figure 2.3(c) has length four and no chords.

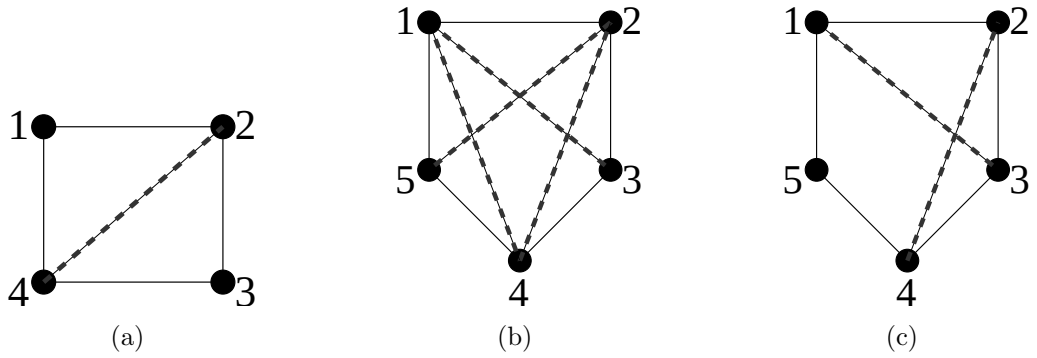


Figure 2.3: Examples of chords, chordal graphs, and one non-chordal graph.

We now define graphs of bounded treewidth by defining a tree decomposition and the treewidth of a graph.

Definition 2.1. *A* tree decomposition *T of a graph $G = (V, E)$ is a tree with the following properties:*

1. *Each vertex $u \in T$ has an associated set V_u of vertices in G.*
2. *$\bigcup_{u \in T} V_u = V$ (i.e. each vertex of G occurs somewhere in T).*
3. *For each $(v_1, v_2) \in E$, there is some $u \in T$ with $v_1, v_2 \in V_u$ (i.e. each edge of G is contained within some vertex of T).*
4. *For each $v \in V$, the subgraph of T induced on $\{u \in T \mid v \in V_u\}$ is connected.*

A graph G is said to have *treewidth* at most k if it has a tree decomposition T such that for each vertex $u \in T$, the maximum size of the set V_u associated with u is $k + 1$.

The last graph theoretic notion that is important for results in Chapter 4 is isomorphism. Graph $G_1 = (V_1, E_1)$ is said to be *isomorphic* to graph $G_2 = (V_2, E_2)$ if there exists a bijective function f mapping V_1 to V_2 such that $\{v_1, v_2\} \in E_1$ if and only if $\{f(v_1), f(v_2)\} \in E_2$. We call the bijective function f an *isomorphism* of G_1 and G_2.

2.2 Allocation Mechanisms

An allocation mechanism considers the following problem: What is the best way to allocate items to one or more individuals? Later, we analyze a particular allocation mechanism known as an auction in great detail, but first we review the underlying allocation problem.

In this section, we introduce the necessary technical details of allocation mechanisms and use these details to define auctions in Section 2.3. While our notation remains consistent with most of the literature, we note that this notation varies somewhat from source to source. We note that sometimes the best way to allocate an item is to allocate it to the system, which we defined in Section 1.1. The system itself is technically also an agent (usually denoted as agent 0, although we assume the agent exists within the system implicitly.) In an auction, agent 0 would represent the seller. For simplicity, we refer to the allocation of an item to the seller as *not allocating* the item. A mechanism involves a set of agents $A = \{1, 2, 3, \ldots, |A|\}$, who communicate with the system in order to achieve some goal.

We would like to introduce the Revelation Principle, which is a theorem, and two properties: incentive compatibility, and individual rationality. The Revelation Principle helps simplify the process of mechanism design. Incentive compatibility and individual rationality help ensure that agents tell the truth and are willing to participate in the mechanism, respectively. We often try to implement these properties either directly from a mechanism, or by restricting certain aspects of the mechanism itself. We formally define a mechanism and the agents involved in the mechanism before presenting the Revelation Principle, incentive compatibility, and individual rationality. In our definitions, we introduce many assumptions that will hold throughout the remainder of this thesis. We summarize these assumptions near the end of this section before working through an example of an allocation mechanism.

In general, a mechanism $\mathcal{M} = (\mathcal{S}_1, \ldots, \mathcal{S}_{|A|}, g, \mathcal{OC}_1, \ldots, \mathcal{OC}_{|A|})$ specifies non-empty sets $\mathcal{S}_i$, one per agent, an output function g, and sets of possible outcomes $\mathcal{OC}_i$, one per

agent. We denote $\mathcal{OC} = \mathcal{OC}_1 \times \cdots \times \mathcal{OC}_{|A|}$. Each non-empty set $\mathcal{S}_i$ is called a *strategy set*, which defines the type of information agent i may communicate to the system. The output function $g : (\mathcal{S}_1, \ldots, \mathcal{S}_{|A|}) \mapsto \mathcal{OC}$ maps the strategy sets to an outcome $OC = (OC_1, \ldots, OC_{|A|})$, where $OC \in \mathcal{OC}$.

Each agent i has some information that is known to the system, and some information that might be known only to the agent. The *type space* Θ_i of agent i describes the agent's possible private information, which is defined by agent i. Typically, the type space Θ_i describes all possible private information about the preferences agent i can have over each possible outcome $OC_i \in \mathcal{OC}_i$. We define the *strategy* of agent i to be a function $s_i : \Theta_i \mapsto \mathcal{S}_i$, which maps the agent's type space onto their strategy set. Each agent i constructs some strategy s_i, subject to $s_i : \Theta_i \mapsto \mathcal{S}_i$. We assume that each agent i knows his or her *true type* $\theta_i \in \Theta_i$, but that their true type is not necessarily communicated to the system and it is not known by any other agent. Instead, agent i communicates their *reported type* $s_i(\theta_i)$ to the system, which may differ from their true type. Each reported type $s_i(\theta_i)$ is available to the system but not necessarily to other agents.

Agent i has a *utility function* u_i, which can be any arbitrary function that is typically known to the system [35, 32]. As is common throughout the literature [35, 32, 46, 42], we assume that agent i is *rational*, which means that they define strategy s_i to maximize the expected value of their utility function. To maximize the expected value of their utility function, an agent must determine the probability of the occurrence of each outcome. Because the details of defining strategies and calculating the probabilities of outcomes are beyond the scope of this thesis, we refer the reader to Mas-Colell *et al.* for more details [35]. As is also common in the literature, we restrict agent i's utility function to the form

$u_i(OC, s_i(\theta_i)) \mapsto \mathcal{R}$, where the agent's type is θ_i [35, 32]. Although such restricted u_i can be any function of inputs OC and $s_i(\theta_i)$ that results in a real number, we often only consider quasi-linear utility functions $u_i(OC, s_i(\theta_i)) = V_i(OC, s_i(\theta_i)) - P_i$, where $V_i : \mathcal{OC}_i \times S_i \mapsto \mathcal{R}$ can be any function, and *monetary payment* P_i is linear and is usually specified by the system as a function of the outcome, $\mathcal{OC}$ [35, 32]. Quasi-linear utility functions are often the only types of utility functions that allow a mechanism to implement some economic properties which we require [35, 32]. Thus, we assume that the utility functions of our agents are quasi-linear. When each agent's utility function is assumed to be quasi-linear, we say that our mechanism is in a *quasi-linear setting*. Quasi-linear utility functions are also referred to as "natural" utility functions [35, 32]. We will see later in this section how monetary payments can be used by special mechanisms to ensure that agents report their true types.

For brevity, we denote a *profile* of the agents' types by $\theta = (\theta_1, \ldots, \theta_{|A|})$. Further, we let $\Theta = (\Theta_1, \ldots, \Theta_{|A|})$.

In some definitions to follow, we will use the term "equilibrium" with respect to agents' strategies in a given mechanism. There is no single definition of equilibrium that is universally agreed upon as appropriate in all situations [35, 32]. A mechanism $\mathcal{M}$ may have more than one equilibrium, depending on its definition, and there may be multiple definitions of equilibria, all of which result in at least one equilibrium for $\mathcal{M}$. For a mechanism $\mathcal{M}$, we say that the agents' strategies are an *equilibrium strategy set* for $\mathcal{M}$ if the strategies satisfy the definition of an equilibrium, and we say that a mechanism *has an equilibrium* if there exists an equilibrium strategy set. The numerous definitions of equilibria are beyond the scope of this thesis, and we refer the reader to Krishna [32] and Mas-Colell *et al.* [35]

for more details. For our purposes, the exact choice of the definition of equilibrium is irrelevant.

A *social choice function* is a function $f : \Theta \mapsto \mathcal{OC}$ that, for each possible profile of agent types $(\theta_1, \ldots, \theta_{|A|})$, assigns an outcome. A mechanism $\mathcal{M}$ *implements* social choice function f if there is an equilibrium strategy set $(s_1^*, \ldots, s_{|A|}^*)$ such that $\mathcal{O}(s_1^*(\theta_1), \ldots, s_{|A|}^*(\theta_{|A|})) = f(\theta)$ for all $\theta = (\theta_1, \ldots, \theta_{|A|}) \in \Theta$.

As we will see in Section 2.3, we are interested in mechanisms that implement social choice functions that are social welfare maximizing, which we now define. A *social welfare maximizing function* is the following [35]:

$$\max_{OC \in \mathcal{OC}} \Sigma_{i=0}^{|A|} u_i(OC, s_i(\theta_i))$$

Because we do not know an agent's true type, we maximize using the agent's reported type, $s_i(\theta_i)$. Social welfare maximizing functions are used because they allow us to associate additional meaning to the chosen outcome of the mechanism. Often, we want to know how an outcome affects the social welfare of the agents, and by implementing a social welfare maximizing function, we can guarantee that the social welfare of the agents is maximized (assuming that the output function can be computed optimally).

Since each agent's utility function is assumed to be quasi-linear, it is known that we can ignore the linear P_i values and simply maximize $V_i(OC, s_i(\theta_i))$ [35, 32]. We now explain this in detail. We note that the summation in the above social welfare maximizing function begins with $i = 0$ instead of $i = 1$. We recall from the beginning of this section that the system is also referred to as agent 0. Agent 0 receives payment P_i from each agent, while

the P_0 portion of their utility is assumed to be 0. Thus, the payments P_i would cancel out in the summation we are attempting to maximize. Therefore, we may ignore agent 0 and simply implement the following social welfare maximizing function:

$$\max_{OC \in \mathcal{OC}} \Sigma_{i=1}^{|A|} V_i(OC, s_i(\theta_i))$$

To try and find all social choice functions, and social welfare maximizing functions, that can be implemented by a mechanism seems to require that we consider all possible mechanisms. Fortunately, the Revelation Principle, which we now present, states that we can restrict attention to mechanisms where $S_i = \Theta_i$. A mechanism $\mathcal{M} = (\mathcal{S}_1, \ldots, \mathcal{S}_{|A|}, g, \mathcal{OC})$ is a *direct mechanism* if $\mathcal{S}_i = \Theta_i$ for all $1 \leq i \leq |A|$. The *Revelation Principle* states that, given a mechanism that implements social choice function f and an equilibrium for that mechanism, there exists another direct mechanism $\mathcal{M} = (\mathcal{S}_1, \ldots, \mathcal{S}_{|A|}, g, \mathcal{OC})$ with $\mathcal{S}_i = \Theta_i$ in which (1) equilibrium outcomes are the same as in the given equilibrium of the original mechanism and (2) $g(\theta) = f(\theta)$ for all $\theta \in \Theta$ [35, 32]. The restriction to direct mechanisms is useful because we need not consider constructing mechanisms with complicated strategy sets in order to implement our social choice function. Instead, we construct a direct mechanism that allows the agents to report their types to the system directly. We assume that our mechanism is a direct mechanism, since we know that such a variation on any mechanism exists.

We note that employing a direct mechanism does not necessarily mean that agents report their type truthfully. If agent i has type $\theta_i \in \Theta_i$, $s_i(\theta_i)$ does not necessarily have to be equal to θ_i as long as $s_i(\theta_i) \in \Theta_i$. In a quasi-linear setting, a direct mechanism

can adjust the monetary payments of each agent's utility function to motivate agents to reveal the truth. These payments can be designed in a way that ensures that rational agents always communicate their true types [35, 32]. The notion of truth is embodied in the concept of incentive-compatible mechanisms, which we now define.

A direct mechanism $\mathcal{M}$ that implements social choice function f is *incentive compatible* if the mechanism $\mathcal{M} = (\Theta, f, \mathcal{OC})$ has an equilibrium $(s_1^*, \ldots, s_{|A|}^*)$ in which $s_i^*(\theta_i) = \theta_i$ for all $\theta_i \in \Theta_i$ and all $1 \leq i \leq |A|$; that is, if each agent telling the truth constitutes an equilibrium of $\mathcal{M}$, then f is incentive compatible [35, 32].

A *Vickrey-Clarke-Groves* (VCG) mechanism is a special mechanism that, in a quasi-linear setting, is incentive compatible while allowing one to implement a social welfare maximizing function [51, 6, 19]. The details of a VCG mechanism are beyond the scope of this thesis. For quasi-linear utility functions, a VCG mechanism is almost always the only mechanism that maximizes social welfare and is incentive compatible [51, 6, 19, 32, 42]. For our purposes, we require a mechanism that maximizes social welfare and is incentive compatible. However, how these two requirements are met does not affect our analysis or results in Chapters 3, and 4.

The last concept we introduce is that of individual rationality. A mechanism in which the amounts agents must pay are so high that an agent is better off not participating at all will not attract any agents. We want agents to participate, and hence we would like the expected value of each agent's utility function to be at least 0. A mechanism that is designed such that the expected value of each agent's utility function is at least 0 is said to be *individually rational*.

The three main concepts discussed above are important because they ensure that agents

are able to simply report their types directly without any extra work, that agents report their true types to the system, and that agents want to participate. The second-to-last point is very important if we attempt to ensure the optimality of the answers within the mechanism. The concept of optimality within our mechanism can also affect how agents report their types.

The output function of a mechanism requires a computation, which may be extremely difficult and perhaps cannot be solved precisely; in this case, an approximation must be obtained or heuristics must be used, or both. Nisan and Ronen show that if agents cannot be given enough of a guarantee of the outcome (i.e. we cannot compute the outcome optimally), then we may no longer have a mechanism that is incentive compatible [42]. It is possible that we lose the ability to implement our social choice function. In this case, our mechanism may also fail to be individually rational [42]. Hence, it is often very important that we are able to solve the output computation exactly, rather than approximating it or using heuristics to find an approximate solution.

Later, when we introduce combinatorial auctions and look for conditions under which they can be solved optimally, we are doing so because of the problems just discussed. Without an exact solution we may lose some very desirable mechanism properties [42]. Namely, we always want incentive compatibility and individual rationality to hold. In an attempt to find conditions where a combinatorial auction can be solved optimally, we place constraints on each agent's strategy set $\mathcal{S}_i$. This can also be tricky, because if we restrict agents' strategies too much, and they are not able to report their types as desired, we may also lose incentive compatibility and individual rationality. Therefore, the restrictions we later place on combinatorial auctions are, for the most part, only useful in specific

situations where they occur naturally, to avoid losing these two mechanism concepts.

Before giving an example of an allocation mechanism, we review the assumptions made throughout this section. For each agent i, we assume that the agent knows his or her true type $\theta_i \in \Theta_i$, and that θ_i is not known by any other agent. We also assume that each agent i is rational and defines strategy s_i to maximize the expected value of their utility function, which is assumed to be quasi-linear. We assume that the system is agent 0 and exists within the mechanism implicitly. Finally, we assume that our mechanism is a direct mechanism.

We now look at a brief example of an allocation mechanism that determines the best way to allocate a pool to a community. In this example, if the pool is allocated then it is allocated to the community, and not to an individual agent. We note that under some conditions, which we will discuss, it may be best to allocate the pool to the system itself. In this case, the pool is said to not be allocated to the community, as we noted at the beginning of this section. Further, we note that in this particular example, allocating a pool to the community is equivalent to *constructing* the pool. Thus, the pool is either built for the whole community, or it is not built at all. We consider a housing community that wishes to construct a community pool for everyone to use. The pool costs \$300, say, and there are three families (agents) within the community. Each family is expected to submit a value to an unbiased party (perhaps a voted community representative), where the value is at least 0. This value is supposed to indicate how much they would be willing to contribute to the construction of the pool, and how much they would value a pool in their community. If the total submitted is less than 300, nothing happens and the pool is not built. If the total is at least 300, then the pool is built and everyone is expected to

pay 100. Each agent i has type θ_i, where θ_i is defined as follows:

$$\theta_1 = 150$$

$$\theta_2 = 125$$

$$\theta_3 = 350$$

We define the utility function of agent i as $u_i = \theta_i - P_i$ if the pool gets built, and $u_i = 0$ otherwise. To begin with, we let $P_i = 100$, for all i and discuss why this is not incentive compatible. We then adjust P_i, for all i, as discussed earlier to ensure that agent's report their true types. To see why the mechanism is not incentive compatible, we consider the case when there is an agent that wishes to have the pool built. As it stands, it would be best for such an agent to submit a type of at least 300 to ensure the pool is built, even if their actual type is less. We need to adjust P_i, for all i, in order to ensure that the mechanism is incentive compatible. To see how this is done, we detail one of the components of a VCG mechanism. We let our social choice function be the social welfare maximizing function that maximizes the sum of the utilities of the agents. To make the mechanism incentive compatible we introduce what are known as Clarke taxes [6].

First, we note that if agent 3 had not participated in the mechanism (i.e. submitted a type of 0), the pool would not have been built. Hence, the utilities of agents 1 and 2 would have been 0, had agent 3 not participated. However, if agent 1 had not participated in the mechanism, but both agents 2 and 3 participate, the utilities of agents 2 and 3 would have been the same because the pool would still have been constructed. The utility value for agent j if agent i had not participated, $i \neq j$, is known as agent i's *marginal effect* on agent

j. The idea of utilities changing based on an agent not participating is used to define Clarke taxes. For agent i, the *Clarke tax* $t_i = [\Sigma_{j \neq i}(u_j)] - [\Sigma_{j \neq i}(u_j \text{ if agent } i \text{ had not participated})]$ [32]. Thus, in our example, the taxes are as follows:

$$t_1 = [(125 - 100) + (350 - 100)] - [(125 - 100) + (350 - 100)] = 0$$
$$t_2 = [(150 - 100) + (350 - 100)] - [(150 - 100) + (350 - 100)] = 0$$
$$t_3 = [(150 - 100) + (125 - 100)] - 0 = 75$$

The Clarke taxes are applied to the mechanism by adjusting P_i, for all i. Specifically, in our example P_i becomes $100 + t_i$, for all i. Therefore, agent 3's utility would decrease by 75. We also noted earlier that such monetary payments are collected by the system. Thus, agent 3 would pay an additional \$75 to the system. If agent i submits their type to the system secretly, so that no other agent knows agent i's type, then the mechanism is an example of a Vickrey-Clarke-Groves mechanism [51, 6, 19, 32]. Therefore, the mechanism is incentive compatible and maximizes the sum of the agents' utility functions.

2.3 Auction Theory

As stated in Section 2.2, an auction is an example of an allocation mechanism. Here we review the notation introduced in Section 2.2 in the language of auctions. We later require the reader to have a working knowledge of the concepts of auction theory, and so we spend some time introducing the various concepts and notation involved.

One or more sellers wish to auction off some items to one or more agents. We denote

the set of items to be sold by I, and refer to a subset $S \subseteq I$ as a *bundle of items*. The agents of the mechanism are referred to as the *bidders*, and are denoted by the set $A = \{1, 2, 3, \ldots, |A|\}$. Each bidder is assumed to know how much they value each bundle of items, which is equivalent to every agent knowing his or her type. Each bidder i's type $V_i = \Theta_i$ is referred to as his or her *valuation*, and we denote the complete set of all valuations by $V = \{V_1, V_2, \ldots, V_{|A|}\}$. We note that V in auction theory is equivalent to $\Theta_1 \times \cdots \times \Theta_{|A|}$ from Section 2.2 on mechanism design. Further, we let $V_i(S)$ denote how much bidder i values the bundle of items S. Where it is clear what bidder we are referring to, we omit the subscript and simply write $V(S)$.

The communication $s_i(\theta_i)$ received from bidder i is known as bidder i's *bid*, which we denote as B_i. The communication process is also known as *bidding* or *placing bids*. We denote the complete set of all bids as $B = \{B_1, B_2, \ldots, B_{|A|}\}$. Further, we let $B_i(S)$ denote the bid placed by bidder i on the bundle of items S. Where it is clear what bidder we are referring to, we omit the subscript and simply write $B(S)$. If $S = \{item_1\}$ we say that $B(S)$ is a bid on item $item_1$. Throughout the literature, this type of auction where bidders are allowed to place bids on any number of bundles of items is referred to as a *combinatorial auction* (CA) [1, 14, 40, 41, 8, 43, 46]. In Chapters 3 and 4 we focus entirely on combinatorial auctions.

A mechanism's output function determines a winner for each item, if a winner exists. The bidder that the output function dictates should receive an item is called the *winning bidder* of that item. In some cases, if there are no bids of sufficiently high value placed on an item (e.g. all bids on an item are less than some threshold), then the seller may opt to keep the item. In this case, no bidder wins the item. We use a social welfare maximizing

function as our social choice function, as is standard in auctions [35, 32]. That is, we want items to be allocated to specific bidders such that social welfare is maximized.

Informally, our social choice function chooses from all possible combinations of mutually disjoint bundles of items, a specific set of mutually disjoint bundles of items $S_{i_1}, S_{i_2}, \ldots, S_{i_j}$ to allocate to bidders $i_1, i_2, \ldots, i_j$, respectively, such that $\sum_{k=1}^{j} B_{i_k}(S_{i_k})$ is maximized. Since we assume that our agents are using quasi-linear utility functions, this function is social welfare maximizing. We present a formal treatment of the social choice function in Section 2.3.1. From Section 2.2, we know that a Vickrey-Clarke-Groves mechanism can implement our social welfare maximizing function and be incentive compatible. Subject to the computational and constraint issues mentioned in Section 2.2, we would also like our incentive-compatible mechanism to be individually rational. We discuss this topic in more detail in Section 2.3.1. The process of determining the winner(s) (i.e. calculating the output of the mechanism) is known as the *winner determination problem* (WDP). Solving the winner determination problem is equivalent to implementing and computing the social choice function. We discuss the winner determination problem in detail in Section 2.3.1.

We assume that the mechanism is designed and implemented by an individual seller. It is possible to extend auctions to multiple sellers easily by remembering which items are sold by which sellers, and then translating the payments from bidders back to the appropriate sellers.

Bidders, as in the more general mechanism specification, are still assumed to be attempting to maximize their utility functions. We will see examples in Section 2.3.1 where this maximization process involves speculation on what other bidders might bid, and why this can lead to problems. The exact details of the utility function are not important in

subsequent chapters. In our later discussion, we focus entirely on the bids submitted by the bidders, and we do not make use of their utility function in any way.

In general, our interest lies in the computational complexity of algorithms that determine the winner(s) of an auction, while allowing $|I| > 1$. An auction in which $|I| > 1$ is termed a *multi-item auction*, and it is the main focus of our later discussion. With $|I| > 1$, a bidder's bid may involve up to $2^{|I|}$ bundles of items S, with $|S| \geq 1$. There are a few approaches to representing multi-item auctions, some of which make use of multiple single-item auctions as an approximation of a multi-item auction; a *single-item auction* implies that $|I| = 1$. It is an approximation because one cannot bid on any bundle of items S with $|S| > 1$, but must place multiple bids on individual items only. As we mentioned in Section 2.2, approximating auctions can interfere with incentive compatibility and individual rationality. In Section 2.3.1, we discuss two multiple single-item auction methods common to the literature [3, 25, 38, 36, 43, 45], as well as the more optimal combinatorial auction approach.

After introducing combinatorial auctions in full detail in Section 2.3.1, we introduce Nisan's concept of bidding languages. In order to analyze the computational complexity of various solutions to the winner determination problem for combinatorial auctions, we represent a combinatorial auction as a bid graph, which we introduce in Section 2.3.3. The bid graph representation requires that certain assumptions be made about how bids are communicated from the bidders to the seller. Due to the direct relationship between the bid graph and the method of communication between the bidders and the seller, we leave discussion of the bid graph construction until after we have established bidding languages in Section 2.3.2.

In Section 2.3.2, we also show how Nisan builds a communication language that satisfies the assumptions required to represent combinatorial auctions with bid graphs. Further, we verify that our assumptions do not cause the communication cost to become exponential in the number of items, should there exist an alternative best-case communication cost without any assumptions that is polynomial in the number of items. If the communication cost is exponential in the number of items, the cost may dominate the running time of any algorithm that solves WDP using a bid graph.

Finally, in Section 2.3.4, we introduce another graph theoretical representation of combinatorial auctions known as item graphs. While both a bid graph and an item graph can be used to represent the same combinatorial auction, we will see that the two representations are very different. For instance, there can be numerous item graph representations of a single combinatorial auction, some of which may allow for a faster solution to WDP than the rest. We spend some time in Chapter 4 closing the gap between bid graphs and item graphs, in an effort to find a specific relationship between the two representations.

2.3.1 Methods for Auctioning Multiple Items

We discuss three approaches for auctioning multiple items, including sequential single-item auctions, parallel auctions, and combinatorial auctions. The first two methods approximate multi-item auctions through the use of multiple single-item auctions. The last approach is a generic technique for auctioning multiple items, and as such we look at it in more detail than the first two. Sequential and parallel auctions are fairly common and well studied in the literature [25, 36, 3, 38, 45], and we will briefly discuss them as well as their limitations. By reviewing the limitations associated with single-item approximations of multi-item auctions

we hope to better appreciate the motivation behind studying combinatorial auctions.

First, we assume that there exists at least one bidder that values a bundle of at least two items either more or less than the sum of all of the values of the items within the bundle. Specifically, there exists a bidder i and bundle of items $S \subseteq I$, $|S| > 1$, such that $B_i(S) \neq \sum_{itm \in S} B_i(\{itm\})$. If no such bidder exists, then we have a multi-item auction that is equivalent to multiple, single-item auctions. Thus, this assumption is necessary to show the power of combinatorial auctions.

In order to demonstrate a weakness in sequential single-item auctions, and in parallel auctions, we introduce the exposure problem. The *exposure problem* occurs when an agent wants items itm_1 and itm_2 together (and not individually), pays a lot to win itm_1, but does not win itm_2. At this point, the agent has bid aggressively to obtain itm_1, but then did not win itm_2. The agent could have potentially overpaid to win itm_1 in hopes that they would win itm_2, and yet failed to do so. Therefore, the agent potentially pays a lot and does not get what they want.

Sequential single-item auctions, where we auction off one item at a time, do not allow bidders to specify preferences for bundles of items. Since there exists a bidder that has a preference for some bundle of size at least two, there is a bidder who does not have the ability to formulate a strategy for bidding that ensures maximum utility [25, 3, 45]. We let auction j denote the jth auction in the sequence of single-item auctions, and we let item itm_j denote the item being allocated in auction j. In order to bid on item itm_j such that the bidder's utility function is maximized at the end of running all auctions, the bidder may need to speculate on the outcomes of all auctions that are to take place after auction itm_j. This is, of course, not feasible for even a reasonably small number of items,

and because bidders do not have complete information about each other we are left with further uncertainty. Since bidders may not know how to maximize their utility function, we run into economic inefficiencies where bidders may not receive items they want, and may receive items that they do not want [46]. Therefore, because at least one bidder does not know how to maximize their utility function, such auctions are not incentive compatible because of their inability to guarantee an optimal solution. In addition, sequential single-item auctions are prone to the exposure problem.

Parallel auctions are another approach to multiple item auctions, in which all items are open for bidding simultaneously, bids must be submitted during certain periods of time, and all bids are observable by all bidders [25, 38]. As all bids are observable, bids submitted by other bidders signal a probable outcome to an auction and thus the need for speculation is potentially reduced. However, parallel auctions do not offer any real solution to the inefficiencies found in sequential single-item auctions. Bids may be altered throughout the parallel auction until the last possible moment, leading to the same situation as before, where a bidder must speculate on future bids with plenty of uncertainty. Parallel auctions are also prone to the exposure problem. Further, because every bidders' best strategy is to wait for all other bids to be submitted, it is possible that no one will place a bid in hopes that the other bidders bid first. There are variations on parallel auctions that try to reduce the amount of uncertainty and force bidders to place bids, but these pitfalls cannot be completely eliminated [36].

In a modified parallel auction where bids are not observable, a bidder's bid on an item is still heavily influenced by the outcome of the other auctions. This follows directly from the exposure problem, which still occurs in parallel auctions where bids are not observable.

For instance, if a bidder wishes to obtain two items, but either item separately is worthless to the bidder, then they have good reason to be very hesitant when bidding on both items, just in case they end up with only one of the pair. Even if the bidder feels that bidding on the pair of items may lead to a higher personal utility, they will want the sum of their bids for the two items to be lower than their valuation of the two items. Bidding this way reduces the loss incurred by the bidder, should they end up with only one of the two items, however it could significantly reduce the utility of the seller. Thus, even in this modified version of a parallel auction, the potential for necessary speculation on the part of a bidder makes the approach suboptimal for both the bidders and the seller. Thus, we lose incentive compatibility.

The last approach to be mentioned here and then discussed at greater length is that of combinatorial auctions. The CA approach allows bidders to express their valuations without unforeseen repercussions. This is a direct result of every bidder's ability to bid on bundles of items and either receiving all of the items in their bid, or none of them. No bidder has to worry about winning an item that has no value to them, as we saw was possible in both sequential and parallel single-item auctions. This benefit of CAs is the motivation behind their study.

To begin our own investigation we need to look at the winner determination problem for combinatorial auctions in more detail. Recall that the general type of auction where bidders may place bids on any bundle of items $S \subseteq I$ is referred to as a combinatorial auction. We assume for combinatorial auctions that $B(S) \in \mathbb{Z}$, for all bundles of items S, as we are dealing with monetary denominations. Further, if bidder i submits a bid on a bundle of items S, then $B_i(S) > 0$. In other words, if bidder i wishes to bid 0 on bundle

of items S, then they do so by not submitting a bid on bundle of items S. With these assumptions in mind, we are interested in solving the winner determination problem for combinatorial auctions (i.e. computing the social choice function). We note that for CAs, the winner determination problem must not allocate an item to more than one bidder. We now review some definitions to help formally define WDP for CAs.

Sandholm notes that only the highest bid for each bundle of items needs to be considered in order to determine the winners of a combinatorial auction [46]. Thus for all $S \subseteq I$ such that at least one bidder submitted a bid on S, we define $B^*(S)$ as follows:

$$B^*(S) = \max_{B_i(S) \text{ submitted}} B_i(S)$$

We use the values of $B^*(S)$ to determine the winning bidders. We denote the set of all defined $B^*(S)$ values by $B^* = \{B^*(S) \mid S \subseteq I$, there is at least one bid submitted on S by some bidder$\}$. Calculating the allocation of items to bidders can be done by solving a linear program, which is a function of valid outcomes.

Definition 2.2. *A* valid outcome $ValO = \{S_1, S_2, \ldots, S_l\}$ *is a collection of bundles of items, where* $S_j \subseteq I$, $|S_j| \geq 1$ *for all* j *and for every* $S_j, S_k \in ValO, j \neq k$, *we have* $S_j \cap S_k = \emptyset$.

A partial solution to the winner determination problem is the following linear program:

$$\max_{ValO} \sum_{S \in ValO} B^*(S)$$

where $ValO$ is a valid outcome. The calculation yields a collection of disjoint subsets of I

that maximize the sum of the bids. Note that some items may not be included in any of the subsets $S \in ValO$ and a valid outcome does not explicitly state who wins each of the subsets. However, the latter can be determined by giving every bundle $S_j \in ValO$ to the bidder who placed the highest bid for S_j, with ties broken arbitrarily.

Any items that are not included in a valid outcome we consider to be in set S' with a bid of zero. These items are kept by the seller at the end of the auction. This tells us exactly what to do with all items that are up for auction.

Definition 2.3. *An* exhaustive valid outcome *is a valid outcome where every item is included in exactly one subset of the outcome.*

To find a solution to WDP, we need to find the solution to the linear program above, and then augment the outcome with S' in order to make it an exhaustive valid outcome.

As we noted in Section 2.2, it is often very important that we are able to solve WDP for CAs optimally, rather than approximating it or using heuristics to find an approximate solution. Optimality is important for maintaining incentive compatibility and individual rationality. Therefore, despite the fact that WDP for CAs is NP-hard [44], we will be interested only in solving it optimally. In Chapter 3, we analyze the computational hardness of finding solutions to restricted versions of the problem. We aim to analyze WDP from another perspective, using a graph representation of combinatorial auctions. First, we need the notion of atomic bids and OR bids defined in Section 2.3.2, as these concepts are used in both the construction of our graph representation and in interpreting its meaning.

2.3.2 Bidding Languages

In order to allocate goods to bidders, the seller needs to know how much each bidder is bidding for every bundle of items. This leads to a computationally difficult problem since the number of possible bundles is exponential; namely there are $2^{|I|}$ possible bundles. Not only does this make the problem of winner determination hard, but communication between a bidder and the seller can be extremely problematic. It is not practical for sufficiently many items to expect every bidder to submit a bid for an exponential number of bundles of items. In real-world applications of combinatorial auctions, a bidder will value most bundles as zero and will value only a few bundles with positive values. We need a concise way of representing the non-zero bids and to accomplish this we rely on specific bidding languages.

For each language, we show for a single bid in the language the values of $B(S)$ for every bundle of items, S. The idea is that communicating a single bid in the bidding language effectively communicates many non-zero $B(S)$ bids. Ideally, the cost of communicating a bid in the bidding language is significantly smaller than the number of non-zero $B(S)$ bids it conveys. We assumed earlier that bidders do not submit bids if they have a value of 0; however here we are listing those bids of value 0 to illustrate the power of each language, and to show the $B(S)$ bids for all $2^{|I|}$ possible bundles of items.

Definition 2.4 ([41])**.** *A bidder can submit an* atomic bid *as a pair* (S, p)*, where* $S \subseteq I$ *and* p *is the maximum price the bidder is willing to pay for* S*.*

An atomic bid (S, p) represents the following set of $2^{|I|}$ bids: $B(T) = p$ for all T such that $S \subseteq T \subseteq I$, and $B(T) = 0$ otherwise. The concept of a single atomic bid is obviously

not sufficient to represent all possible bids a bidder may wish to express, as a bidder may wish to bid $B(T) > 0$ for some $T \subset S$. To try to remedy this, Nisan extends the notion of an atomic bid to *OR bids*.

Definition 2.5 (OR bidding language [41])**.** *A bidder can submit, as a single* OR bid*, an arbitrary number of atomic bids* $\{(S_1, p_1), (S_2, p_2) \ldots, (S_q, p_q)\}$*. Such a bid is denoted* $ORB = (S_1, p_1)$ *OR* $\ldots$ *OR* (S_q, p_q)*.*

OR bids are a representation of $2^{|I|}$ bids of the form $B(S) = p$, only written in a potentially more concise manner. OR bids relate to our previous bid definition in the following way:

Definition 2.6. *We assume bidder* i *submits an OR bid,* $ORB_i = (S_1, p_1)$ *OR* $\ldots$ *OR* (S_q, p_q)*. Then the bid* $B_i(S)$ *on a bundle* $S \subseteq I$ *is defined to be the maximum over all possible valid collections* $W = \{j_1, \ldots, j_\ell\}, 1 \leq j_k \leq q$*, of the value* $\sum_{j_k \in W} p_{j_k}$*, where* W *is valid if* $S_{j_k} \subseteq S$ *for all* $1 \leq k \leq \ell$*, and* $ValO = \{S_{j_k} \mid j_k \in W\}$ *is a valid outcome. If there exists no* S_{j_k} *such that* $S_{j_k} \subseteq S$*, then* $B_i(S) = 0$*.*

In other words, the bidder is willing to receive entire item bundles from any number of disjoint atomic bids for the cost of the sum of their respective bid values in such a way that maximizes that sum. Unfortunately, OR bids also have a limitation in what they can express; they cannot represent the case where there are two bundles S, T where $S \cap T = \emptyset$, and a bidder wishes to place the bid $B(S \cup T) < B(S) + B(T)$ [41]. Nisan once again extends the notion of an atomic bid; this time to *XOR bids*.

Definition 2.7 (XOR bidding language [41])**.** *A bidder can submit, as a single* XOR bid*,*

an arbitrary number of atomic bids $\{(S_1, p_1), (S_2, p_2), \ldots, (S_q, p_q)\}$. *Such a bid is denoted* $XORB = (S_1, p_1)\ XOR\ \ldots\ XOR\ (S_q, p_q)$.

XOR bids also represent $2^{|I|}$ bids of the form $B(S) = p$. Here, the bidder is willing to receive the bundle of items of at most one atomic bid specified in the XOR bid. XOR bids relate to our previous bid definition in the following way:

Definition 2.8. *We assume bidder* i *submits an XOR bid,* $XORB_i = (S_1, p_1)XOR\ldots XOR(S_q, p_q)$. *Let* $S \subseteq I$ *be a bundle of items. If there is no* $S_j \subseteq S, 1 \leq j \leq q$, *then define* $B_i(S) = 0$. *Otherwise, the bid* $B_i(S)$ *is defined to be* $B_i(S) = \max_{j|S_j \subseteq S} p_j$.

Lemma 2.9 ([41])**.** *XOR bids can represent all possible ways a bidder may wish to place bids.*

Proof. One can clearly represent any bid $B(S) = p$ by an atomic bid (S, p) in an XOR bid. It only remains to see that XOR bids can also represent the case where there are two bundles $S, T, S \cap T = \emptyset$, and we wish to represent the bid $B(S \cup T) < B(S) + B(T)$.

First, we note that $B(S \cup T) \geq \max\{B(S), B(T)\}$, since if $B(S) \geq B(T)$, say, and one were willing to pay $B(S)$ for the bundle S, then one wouldn't care if more items were received at no extra cost. If we were to add on an atomic bid to our XOR bid of the form $(S \cup T, B(S \cup T))$, then by Definition 2.8, we must be willing to pay $\max\{B(S), B(T), B(S \cup T)\}$ for the bundle of items $S \cup T$. This is exactly $B(S \cup T)$, as desired, and thus XOR bids can in fact represent all possible bidding strategies. □

As noted at the end of Section 2.3, we must verify that our assumptions in Section 2.3.3 do not cause communication to become unreasonable in terms of the number of items.

Hence, we are interested in comparing how concise two different bidding languages are, which Nisan does by comparing the size of bids from the respective languages. We are interested in comparing the minimum number of atomic bids necessary to express a desired set of bids, $\{B(S) \mid S \subseteq I, B(S) > 0\}$, within the different bidding languages.

Definition 2.10. *The* size *of a bid is the number of atomic bids in it.*

Nisan gives examples of "natural" valuation functions that for $|I|$ items have linear size in the OR bidding language, but exponential size in the XOR bidding language, and vice-versa. Further, Nisan extends the idea of OR bids and XOR bids to OR-of-XOR bids, and to XOR-of-OR bids. The former allows a bidder to submit an arbitrary number of XOR bids, and the latter allows a bidder to submit an arbitrary number of OR bids. Both have their strengths and weaknesses, which are unimportant to the work presented in this thesis. Further, Nisan considers allowing any number of atomic bids to each be separated by either OR or XOR operators arbitrarily (and of course we must allow the use of brackets to help with order of operations). These are known as OR/XOR bids and the language is fully expressive. Further, the OR/XOR bidding language requires fewer atomic bids to represent a desired set of bids than any other bidding language [41].

Finally, we introduce Nisan's notion of OR* bids in order to overcome the limitation of OR bids, while maintaining a similar relation to our previous bid definition $B(S)$ as was discussed of OR bids. In order to formally define OR* bids, Nisan introduces the concept of dummy items. *Dummy items* are items that have no value to any of the bidders, but they allow us to express exclusivity. Recall that OR bids require a valid collection W in order to relate to bids of the form $B(S)$. If two subsets S_1, S_2 are disjoint and we add dummy item d_0 to both sets, then we have indicated that a bidder wishes the sets to remain exclusive

upon allocation. That is, no valid collection W can contain both 1 and 2, as $S_1 \cap S_2 \neq \emptyset$ once d_0 is added to both sets. More formally, we allow each bidder i to have their own set of dummy items D_i, which only that bidder can bid on.

Definition 2.11 (OR* bids [41])**.** *A bidder can submit, as an* OR* bid*, an arbitrary number of pairs* $\{(S_1, p_1), (S_2, p_2) \ldots, (S_q, p_q)\}$*, where for* $1 \leq \ell \leq q$*,* $S_l \subseteq I \cup D_i$*. Such a bid is denoted* $OR^*B = (S_1, p_1)$ *OR* $\ldots$ *OR* (S_q, p_q)*.*

OR* bids can be handled just like OR bids and it follows that any algorithm which assumes that bidders bid using OR bids will work if bidders were to submit OR* bids instead. OR* bids are fully expressive as they clearly contain the power to represent any XOR bid. Further, Nisan shows that the OR* bidding language is concise. Specifically, any valuation that can be represented by an OR/XOR bid of size s can also be represented by OR* bids of size s using at most s^2 dummy items [40]. This shows that if we assume communication in the OR* bidding language, then our input size is at most a polynomial of the size of the input had the communication been in the OR/XOR bidding language. We conclude that an assumption that bidders communicate bids using the OR* bidding language is acceptable. Thus in later sections, when algorithms we investigate assume an OR bidding structure, we are not pursuing investigations that cause communication to become intractable because we can assume that bidders make use of the OR* bidding language.

2.3.3 Bid Graphs

One possible means for representing a combinatorial auction is that of a bid graph, which will be used in algorithms for solving and analyzing WDP in Chapter 3. Bid graphs have been used previously in the literature, although no direct credit is given for the concept's origin [44, 46, 48]. In this section we present a complete and formal description of a bid graph, as it will be examined extensively in Chapters 3 and 4.

First, note that we assume bids are submitted using an OR bidding structure as discussed in Section 2.3.2, which we assume is accomplished using the OR* bidding language.

Each vertex in a bid graph is a bid, and there is an edge between two vertices if the two bids they represent share an item. Recall that A is our set of bidders, and I our set of items. We denote the bid for bidder i as B_i, which consists of atomic bids $(S_{i1}, p_{i1}), \ldots, (S_{i|B_i|}, p_{i|B_i|})$ where $S_{ij} \subseteq I$ and $p_{ij} > 0$. For each bidder i, we define $dab(B_i)$ to be the number of distinct atomic bids in B_i. Lastly, we let $dab(B)$ represent the number of distinct atomic bids across all B_i; we distinguish between two atomic bids by their item bundles. The number of distinct atomic bids is not simply a summation over i of $dab(B_i)$ since agents may bid on the same bundles of items. However, $dab(B) \leq \sum_{i=1}^{|A|} dab(B_i)$. Equivalently, $dab(B)$ represents the number of distinct bundles of items $S \subseteq I$ such that some bidder has submitted a bid for S. We give the following formal definition of a bid graph:

Definition 2.12 (Bid Graph)**.** *Given a combinatorial auction with set of bidders A, set of items I, and bids B, a* bid graph $G = (V, E)$ *for the auction must satisfy the following:*

1. $V = \{S \mid S \subseteq I$ *and some bidder placed a bid on* $S\}$ *(We note that* $|V| = dab(B)$*).*

2. *Each vertex $v \in V$ is labeled $\{i, S, p\}$, where $S \subseteq I$ is a bundle of items, i is the bidder who placed the highest overall bid on bundle S (with ties broken arbitrarily), and $p > 0$ denotes the value of that bid.*

3. *For any two vertices $v_1 = \{i_1, S_1, p_1\}, v_2 = \{i_2, S_2, p_2\} \in V$, the following holds:*

   - *$\{v_1, v_2\} \in E$ if and only if $S_1 \cap S_2 \neq \emptyset$.*

We now construct a bid graph for all bidders that satisfies Definition 2.12. Our algorithm will consist of three parts: the first will remove duplicate bids of less value, the second will construct and maintain a data structure for constructing edges in the bid graph, and the third will actually construct the bid graph. For our construction we assume that each bid is on a bundle of items S that is encoded via an array of $|I|$ bits. Bit i is 1 if and only if item i is included in bundle of items S.

First, we note that our vertices are a set, and thus contain no duplicates. This is similar to the definition of set B^*, and in fact $|V| = dab(B)$, the number of distinct atomic bids. Thus, each time we receive a bid on a bundle of items we have previously seen bid on by another agent, we will need to ignore this bid unless its value is greater. This can clearly be done by maintaining a sorted list of bids, where each bid points to its vertex in the bid graph and if we find a duplicate, then we update the vertex if needed with the potentially higher bid. However, there is a faster way of removing duplicates. By maintaining a sparse binary tree, we can treat each bid encoding as a path from the root of the tree to a leaf of length exactly $|I|$. We begin by considering bit 1 of each bid, and starting at the root of the sparse binary tree. For each bid, we read the current bit and if it is 1 we follow the path to the current node's right subtree, and if the bit is 0 we follow the path to the

current node's left subtree. If we find part of the path is missing, we fill it in. At each leaf, we store a pointer to the vertex that represents the bid, creating one if necessary. The size of the tree is then at most $O(dab(B) \cdot |I|)$. To construct the tree, we need to consider each incoming bid, of which there are exactly $M = \sum_{i=1}^{|A|} |B_i|$. For each bid, we need to do exactly $|I|$ work decoding the bid and following its corresponding path down the binary tree. Thus, removing duplicates takes exactly $M|I|$ time.

We note here that if a bid on a bundle of items has previously been seen, then the leaf reached during the above process will point to a vertex. We would then verify in constant time whether or not we need to update the vertex with a new bid value and agent number. Only if the leaf did not exist before, and this is the first time we have seen a bid on this precise bundle of items do we pass the vertex associated with the bid on to the second part of the algorithm.

The next structure we build aids in constructing the edges of the bid graph, and the part of the algorithm we describe now takes vertices of the final bid graph as input, which are passed from the first portion of the algorithm introduced above. In this part of the algorithm, we maintain a sparse matrix where the rows of the matrix are numbered $1, \ldots, |I|$ and represent the items. Each row i is filled by a linked list of pointers to vertices, which represent all distinct bids currently submitted that contain i. For example, if agent 4 bids on the bundle of items $\{1, 5, 7\}$ then rows 1, 5 and 7 of the matrix would add a pointer to the vertex associated with this bid to its linked list. We note that the bids considered here have already made it through the duplication removing process. Thus, each vertex seen by this part of the algorithm is assumed to be the first and only vertex seen on its exact bundle of items. After having seen all bids by all agents, each row i of the matrix is

a linked list containing pointers to the vertices of all of the distinct atomic bids on bundles of items containing i.

The maximum size of the sparse matrix is $O(dab(B)|I|)$, and takes exactly $O(dab(B)|I|)$ time to construct; each vertex requires $O(|I|)$ time to insert all of the required pointers into the sparse matrix. However, before the pointers to the vertex are inserted into the sparse matrix, we add edges to the bid graph. At this point in our description, we have only described the construction of the vertices, which happens during the removal of duplicates. We now explain how the appropriate edges are added to the bid graph.

We let $deg(v)$ be the *degree* of vertex v. Then, for each vertex v, and for each item i in v's bundle of items, we iterate over the list of pointers in the sparse matrix in row i. This row has at most $deg(v)$ pointers in it. For each pointer to vertex u, we add an edge between u and v (which must not exist) because u also represents a bid that contains item i, by construction. The edges necessary can be added in $O(deg(v) \cdot |I|)$ time, in this fashion. This process needs to be done $dab(B)$ times, for each newly constructed vertex. However, the sum over all vertices of the degree of each vertex is exactly twice the number of edges [10]. Therefore, this part of the algorithm constructs the edges of the bid graph in time $O(|E| \cdot |I|)$, where $|E|$ is the total number of edges in the completed bid graph.

To summarize, our algorithm for constructing the bid graph takes time $O(M|I| + dab(B)|I| + |E| \cdot |I|)$ time to complete, where $M = \sum_{i=1}^{|A|} |B_i|$ and $|E|$ is the total number of edges in the completed bid graph. The vertices in our bid graph are the complete set of distinct atomic bids across all bidders, and the graph explicitly shows, via the edges, pairs of bids that cannot both be part of a solution to WDP due to shared items. Embedded in the bid graph is the concept of OR bidding, where we can select any number of vertices

provided that they pairwise do not share any items in common. Our construction allows for a translation to and from a CA in polynomial time. We later analyze the bid graph and consider the ramifications of various restrictions on this graph and its associated CA.

Further, our technique allows construction to occur while agents are bidding, and not merely after all agents have submitted their bids. Thus, the algorithm can be made to run concurrently with the bidding process. As bids are submitted by bidders, the algorithm may process all of those bids while we wait for bids to be submitted by other bidders. In this way, it may be possible to amortize some of the cost as the seller must wait until all bidders have submitted their bids before calculating the winning bidders.

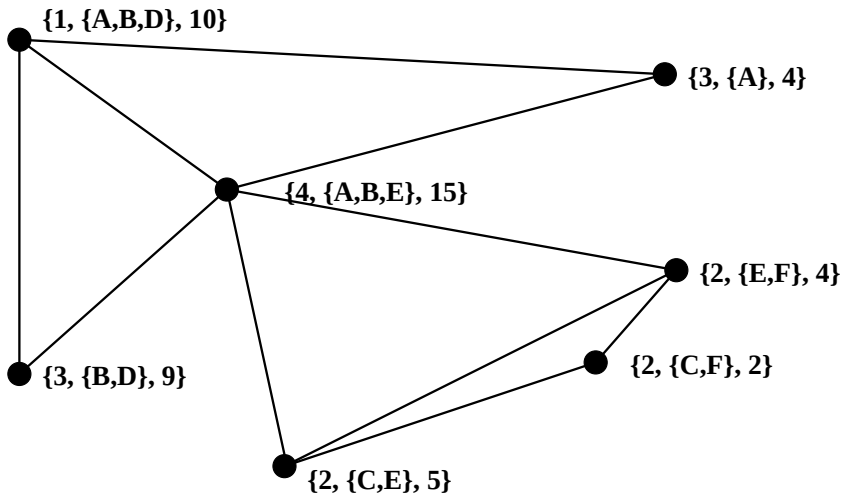


Figure 2.4: An example of a bid graph representation of a CA.

Figure 2.4 is an example of a bid graph on the following bids: $B_1 = (\{A, B, D\}, 10)$ OR $(\{B, D\}, 8)$, $B_2 = (\{C, E\}, 5)$ OR $(\{C, F\}, 2)$ OR $(\{E, F\}, 4)$, $B_3 = (\{A\}, 4)$ OR $(\{B, D\}, 9)$ OR $(\{A, B, E\}, 10)$, $B_4 = (\{A, B, E\}, 15)$. In this example, bidders 1 and 3

both bid on bundle $\{B, D\}$, however bidder 3 bid more, and hence the vertex representing this bundle is labeled $\{3, \{B, D\}, 9\}$. Similarly, bidders 3 and 4 both bid on bundle $\{A, B, E\}$ with bidder 4 submitting a larger bid value of 15. Two bids in the graph are connected if and only if they share at least one item, as previously described. WDP, as defined in Section 2.3.1, is equivalent to finding a maximum weighted independent set on the constructed graph, where we define the weight of any vertex $v_\ell = \{\ell, S_\ell, p_\ell\}$ to be p_ℓ.

2.3.4 Item Graphs

Item graphs are a representation of combinatorial auctions altogether different from the bid graphs we introduced in Section 2.3.3. We introduce item graphs in this section, and in Chapter 4 show how they can be used to solve WDP. Similar to our use of the bid graph representation, we will be interested in restricting our attention to combinatorial auctions that can be represented by an item graph with a specific structure, in order to obtain a solution to WDP with better than exponential running time. In addition, we will be interested in how any such restriction to a combinatorial auction affects the CA's bid graph, if at all. For example, we may wish to restrict our attention to CAs with item graphs that are paths, or trees, and it may be the case that these CAs have very specific structure in their bid graphs. Of special interest will be the fact that given an item graph, it is not immediately obvious what combinatorial auction it represents, and as such what the corresponding bid graph may be. This will obviously affect our ability to perform the above analysis.

Once again, note that we assume bids are submitted using an OR bidding structure as discussed in Section 2.3.2, which we assume is accomplished using the OR* bidding

language.

Informally, in a valid item graph of the combinatorial auction, the bids must be connected induced subgraphs of the item graph. Formally, we have the following definition:

Definition 2.13. *Given a combinatorial auction, a* valid item graph $G = (I, E)$ *representing the given CA must satisfy the following conditions: each item is represented by exactly one vertex in the graph, and for each atomic bid* (S, p)*, the induced subgraph of* G *on the vertices contained in* $S \subseteq I$ *must be a connected graph.*

For example, let's assume we are working with a combinatorial auction on four items consisting of the following atomic bids: $(\{1\}, 1)$, $(\{1, 2, 3\}, 5)$, $(\{2\}, 4)$, and $(\{2, 3, 4\}, 6)$. For simplicity, we assume that we have only one bidder. The graphs in Figure 2.5 are all examples of valid item graphs for this combinatorial auction. We note how in each example there are exactly four vertices (corresponding to the total number of items in the CA). Further, for each example, if we restrict our attention to the vertices corresponding to any particular atomic bid, then the subgraph induced on those vertices is connected. For comparison, the bid graph of the combinatorial auction is given in Figure 2.6.

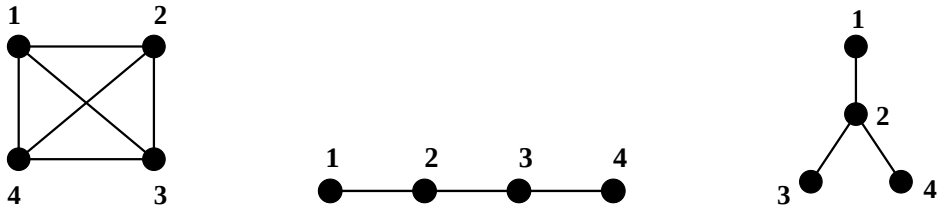


Figure 2.5: Examples of valid item graphs for the same combinatorial auction.

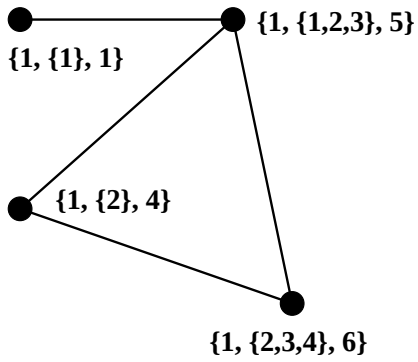


Figure 2.6: An example of a bid graph.

There are other examples of valid item graphs for the above combinatorial auction. Specifically, the addition of edge $\{1, 3\}$ to the item graph in Figure 2.5(b) or 2.5(c) results in two additional valid item graphs for the above combinatorial auction. Further, there are many other combinatorial auctions for which Figure 2.5(a) is also a valid item graph. In particular, any combinatorial auction on four items. Consequently, given an item graph we cannot know for certain which combinatorial auction it represents. In Section 2.3.3, we state that one may translate a bid graph to and from a combinatorial auction and in polynomial time. More importantly, any combinatorial auction obtained from a bid graph yields the same solution to WDP, ignoring the case where multiple agents submitted the same highest bid on a bundle of items. Since an item graph may represent many combinatorial auctions and allows for many different combinations of bids and bid values, the solution to WDP could be different for each CA that the item graph may represent. Obviously then, we cannot formulate a specific bid graph given an item graph. However, we can prove various properties about the bid graph of any combinatorial auction that may result in a given item graph. As mentioned above, this will be one of the goals of

Chapter 4.

Since there are many valid item graphs for a single combinatorial auction, we cannot discuss the running time of the creation of an item graph in simple terms. There may be many different algorithms for creating valid item graphs, which may have different properties from one another. We present some past results pertaining to the hardness of finding various valid item graphs in Chapter 4. For example, we discuss the hardness of finding a valid item graph using the fewest number of edges possible, and also the hardness of finding a valid item graph with a treewidth of one, should one exist.

2.4 Parameterized Complexity

Parameterized complexity theory and the notion of fixed-parameter tractability were developed by Downey and Fellows to classify intractable problems into various categories [11]. Our analysis of combinatorial auctions and the winner determination problem on combinatorial auctions requires that we make use of parameterized complexity to distinguish among intractable algorithms so that we may find algorithms that are (relatively) less computationally hard. In Chapter 3 we will be interested in distinguishing algorithms by their relative hardness in the realm of parameterized complexity theory. As we will see, some algorithms are considered fixed-parameter tractable (easy), while others are fixed-parameter intractable (hard); both concepts we define in full detail below. First we introduce the concept of a parameterized problem.

Definition 2.14 ([11])**.** *A* parameterized problem *is a language* $L \subseteq \Sigma^* \times \Gamma^*$*, where alphabets* Σ *and* Γ *are finite. For each* $(x, y) \in L$*, we call* x *the* input *and* y *the* parameter*.*

Many computation and optimization problems that are known to be NP-complete may be represented as parameterized problems in order to obtain new algorithms and complexity results. Often the parameter y is taken to be a positive integer and denoted instead as k. By separating the problem input into two parts, we hope to find an algorithm that has "good" running time as a function of $|x|$, while allowing for arbitrarily "bad" running time as a function of $|y|$. This leads to the formal notion of *fixed-parameter tractability*, which is central to parameterized complexity in the same way that *polynomial time* is central to classical computational complexity.

Definition 2.15 ([11])**.** *A parameterized problem L is* fixed-parameter tractable *(FPT) if there exists an algorithm that, given input $(x, y) \in \Sigma^* \times \Gamma^*$, can correctly determine if $(x, y) \in L$ using time $f(|y|) \cdot p(|x|)$ for some computable function f and polynomial p.*

Fixed-parameter tractability generalizes polynomial time computability by admitting algorithms whose running time is exponential, but only with respect to the parameter. There are combinatorial problems for which proofs of fixed-parameter intractability exist. However, as with the classical notion of intractability, to demonstrate fixed-parameter intractability for most interesting natural problems, we need to make use of completeness theory. More precisely, we need to define what it means for one parameterized problem to be *reducible* to another. Conceptually, we wish to take a parameterized problem instance and find a mapping to the other parameterized problem instance such that an input is in the first language if and only if the mapped input is in the second language. This mapping algorithm must be fixed-parameter tractable and the part of the mapping that calculates the second problem's parameter must be a function of only the first problem's parameter. More formally, we have the following definition:

Definition 2.16 ([11]). *Let $L \subseteq \Sigma^* \times \Gamma^*$ and $L' \subseteq (\Sigma')^* \times (\Gamma')^*$ be parameterized problems. We say that L is* fixed-parameter reducible *to L' ($L \leq^{fpt} L'$) if there is a mapping $F : \Sigma^* \times \Gamma \mapsto (\Sigma')^* \times (\Gamma')^*$ such that*

1. *For all $(x, y) \in \Sigma^* \times \Gamma^*$, $(x, y) \in L$ if and only if $F(x, y) \in L'$,*
2. *F is computable in time $f(|y|) \cdot p(|x|)$, for some computable function f and polynomial p, and*
3. *For all $(x, y) \in \Sigma^* \times \Gamma^*$ where $F(x, y) = (x', y')$, we have $|y'| \leq g(|y|)$ for some computable function g.*

The mapping F in Definition 2.16 is also referred to as a *fixed-parameter reduction* from L to L'. Fixed-parameter reductions are useful in proving whether or not parameterized problems are FPT. More precisely, we can show that a parameterized problem L is FPT by providing a fixed-parameter reduction from L to another parameterized problem that has previously been shown to be FPT [11]. Further, we can prove that a parameterized problem L is fixed-parameter intractable by providing a fixed-parameter reduction from a known fixed-parameter intractable problem to L [11]. As we will see shortly, there exist a number of parameterized complexity classes that a fixed-parameter intractable problem may belong to. If a fixed-parameter tractable problem L' is hard for some class, $\mathcal{C}$ say, then we can prove that a parameterized problem L is also hard for $\mathcal{C}$ via a fixed-parameter reduction from L' to L [11]. In this case, we say that both L and L' are $\mathcal{C}$-hard. In Chapter 3, we use fixed-parameter reductions to show that some parameterized problems are fixed-parameter tractable.

We now have the tools and definitions to perform fixed-parameter reductions, but it remains to identify classes of problems that are believed to be fixed-parameter intractable. To do this we need to define decision circuits as we will use certain restrictions on these circuits to define classes for fixed-parameter intractable problems. A decision circuit is a boolean circuit of AND and OR gates with an arbitrary number of inputs but only one output. The *fanin* of an AND or OR gate is the number of inputs to the gate.

Definition 2.17 ([11]). *A gate in a circuit is* large *if its fanin exceeds some bound. The* weft *of a decision circuit C is the maximum number of large gates on any path from the input variables to the output. The* depth *of a circuit is the maximum number of gates on any path from the input variables to the output.*

Consider a family of decision circuits $\mathcal{F} = \{C_1, C_2, \ldots, C_n, \ldots\}$. The parameterized language associated with $\mathcal{F}$ is

$$L_{\mathcal{F}} = \{\langle C_i, k\rangle : C_i \text{ has a weight } k \text{ satisfying assignment}\}.$$

We denote the family of weft t depth h decision circuits as $\mathcal{F}_{(t,h)}$, and its parameterized language as $L_{\mathcal{F}_{(t,h)}}$.

Definition 2.18 (Basic Hardness Class [11]). *A parameterized language L is in the class $W[t]$ if and only if L is fixed-parameter reducible to $L_{\mathcal{F}_{(t,h)}}$ for some h.*

$W[t]$ has been defined using the languages for families of decision circuits with bounded depth. A natural question would be to ask what happens if we have no such bound on the circuit depth, but rather a restriction on the overall size of the circuit. Downey and

Fellows allow the circuit size to be polynomial in the number of circuit inputs and define two additional weft classes by specifying the type of circuit involved [11].

Definition 2.19 ([11]). *$W[SAT]$ denotes the class obtained by enforcing a polynomial-size restriction on* boolean *circuits, and $W[P]$ is obtained by enforcing a polynomial-size restriction on* decision *circuits.*

Definition 2.20 (The W-Hierarchy [11]). *The union of the $W[t]$ classes together with $W[SAT]$ and $W[P]$ forms the* Weft-Hierarchy *or* W-Hierarchy*. The relationship amongst these classes can be described as follows,*

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[SAT] \subseteq W[P].$$

For the remainder of this document we will only see parameterized problems belonging to either FPT or $W[1]$. The rest of the hierarchy is useful for classifying the relative hardness of fixed-parameter intractable problems, but will not come up in our discussion of combinatorial auctions. In essence, $W[1]$ will simply denote fixed-parameter intractability for our parameterizations of WDP, and it is no more special than any other $W[t]$ class except that it is believed to be a "simpler" fixed-parameter intractable problem; its relative hardness is believed to be less than that of $W[t]$ with $t > 1$ [11]. In Chapter 3, we provide fixed-parameter reductions from known $W[1]$-hard problems to various parameterizations of the winner determination problem to demonstrate that these parameterizations are $W[1]$-hard. In addition, we show that a parameterization of WDP is $W[1]$-complete via a fixed-parameter reduction from the problem to a known $W[1]$-complete problem.

In the context of this thesis, it is important to know if a parameterization of WDP is

$W[1]$-hard because, as we will see in later sections, $W[1]$-hardness implies that our parameterization does not provide a significant improvement in the running time of a solution to WDP. Knowing which restrictions *not* to make is often just as important as making ones that lead to fixed-parameter tractability. For further details about the hierarchy and its definition, see Downey and Fellows [11]. For the classification of various parameterized problems see Marco Cesati [4].

2.5 The Set Packing Problem

In this section we formally define the weighted set packing problem and show how this problem relates to the winner determination problem for combinatorial auctions. Further, we review previous results that apply to the weighted set packing problem, including results that use parameterized complexity theory. At the end of this section, we introduce the standard method for representing an instance of the weighted set packing as a hypergraph, and how this relates to graph representations for WDP for CAs.

The winner determination problem for combinatorial auctions is an instance of a well-known problem; the weighted set packing problem (w-SPP) [44, 46, 9, 18]. Given I and B, where I is a set of elements and B is a collection of subsets of I and each element of B is assigned a nonnegative weight, *the weighted set packing problem* is to find a set S of pairwise disjoint elements of B such that the sum of the weights of the elements of S is maximized. The decision version of w-SPP asks whether or not the sum of the weights of the elements in S is at least k, and we denote this version of w-SPP as w-SPP(k).

We can see how this easily translates to the winner determination problem for combi-

natorial auctions. The set I is the set of items of the auction, and B is the set of distinct atomic bids across all agents. The elements of B are bids, and the nonnegative weight assigned to each bid is its integer bid value that was submitted by some agent. A solution to the winner determination problem for combinatorial auctions is precisely a solution to the weighted set packing problem with the above definition [46, 9, 18]. Therefore, any algorithm providing a solution to the weighted set packing problem also provides a solution to WDP for CAs, and vice-versa. We now briefly review various results concerning w-SPP.

Using a brute force approach, we can calculate the weight of each subset of B, checking if the elements in the subset are disjoint, in $O(2^{|B|})$ time. For instances of WDP for CAs, this running time is exponential in the number of bids, which may be very large. Thus, the naive approach is impractical in the general case. Furthermore, w-SPP is NP-hard [29].

However, we can view w-SPP from the perspective of parameterized complexity. A uniform-weight decision version of w-SPP asks whether there are k disjoint subsets of I in B. We denote the uniform-weight version of w-SPP as SPP(k). SPP(k) is known to be $W[1]$-complete via reductions to and from independent set with a similar parameterization [11], which we define in full detail in Section 3.1. Further, SPP(k) can easily be reduced to and from w-SPP(k) via a fixed-parameter reduction by reducing to and from independent set. In Section 3.1, we see how w-SPP(k) reduces to and from a parameterization of independent set via fixed-parameter reductions.

The set packing problem, SPP(m, k), where each element of B contains at least m elements, also asks whether there are k disjoint subsets of I in B. SPP(m, k) is also NP-complete, for $m \geq 3$ [15]. Interestingly, Jia *et al.* showed that SPP(m, k) is fixed parameter tractable, and gave an algorithm that uses $O(m^k(g(m,k)^{mk} + k^2m^2|B|))$ time,

where $g(m,k)$ is linear in mk [28].

In SPP(t,k), we look for a set S of k disjoint elements of B so that the total number of elements from I in S is t. Koutis found that this problem can be solved in $O(2^{O(t)}|B||I|\log|I|)$ time [31]. In addition, Koutis presents a randomized algorithm that solved SPP(t,k) in $O(e^t(t|B||I| + t^2 4^t \log k))$ time [31]. Koutis also shows how to solve SPP(k) in $O(k|B|^2|I|^2 2^{|I|})$ time [31].

There are special cases in which SPP(k) can be solved in polynomial time [9]. We can formulate SPP(k) as a linear program, and under certain conditions the program can be solved in polynomial time. We consider ordering the sets in B as $B_1, \ldots, B_n$ and let A be a binary matrix where $A_{ij} = 1$ if an only if the j^{th} set contains element $i \in I$. With this definition of matrix A, SPP(k) can be solved in polynomial time when A satisfies certain properties. For instance, if each column of A contains at most two 1s, then the problem can be reduced to maximum weight matching in a graph, which is solvable in polynomial time [9]. There are many other special cases in which SPP(k) can be solved in polynomial time. Since WDP for CAs is an instance of w-SPP, then the decision version of WDP for CAs is exactly w-SPP(k). Stated previously, w-SPP(k) has a fixed-parameter reduction to SPP(k), and thus if the decision version of an instance of WDP is formulated as SPP(k) and satisfies some set of constraints which lend themselves to a polynomial time algorithm, then we can solve the decision version of WDP in polynomial time. As we will see in Section 3.1, we denote the decision version of WDP for CAs as WD(k), which is a parameterized version of WDP for CAs.

A *hypergraph* is a graph $H = (V, E)$, where each edge is a set of two or more vertices. Interestingly, for any given instance P of w-SPP with elements I and collection B, which

is a collection of subsets of I, the standard graph theoretical representation for P is a hypergraph H whose vertices are the elements I and whose edges are exactly the set B, where each edge $B_i \in B$ is assigned the non-negative weight of its corresponding set from the given instance of w-SPP. We denote such a hypergraph representation of P as $H = (I, B)$. The standard intersection graph of a hypergraph H is a graph $G = (B, E)$ whose set of vertices is exactly B, the set of edges of H, with the weight of a vertex $B_i \in B$ equal to the weight of edge B_i from H, and whose edges are defined as follows:

$$\{B_i, B_j\} \in E \iff B_i, B_j \in B \text{ and } B_i \cap B_j \neq \emptyset$$

We recall that both B_i and B_j are subsets of I.

The definition of the standard intersection graph of a hypergraph H representing an instance P of w-SPP (and consequently an instance of WDP for CAs) is exactly the definition of a bid graph, which we defined in Section 2.3.3, where $B = B^*$. The hypergraph representation of the winner determination problem is used extensively by Gottlob and Greco [18]. They also define the notion of a strict hypertree decomposition and strict hypertree width. Using these definitions, they prove there is a relationship between the dual of the standard hypergraph representation of a CA and item graphs of bounded treewidth. We let $H = (I, B)$ be a standard hypergraph representation of a combinatorial auction CA, such that if $i \in I$ then $\{i\} \in B$. Then, Gottlob and Greco show that a valid item graph of treewidth at most k exists for CA if and only if the dual of H has a hypertree decomposition of width at most $(k + 1)$ [18]. While not a direct relationship between bid graphs and item graphs, the result by Gottlob and Greco shows a definite

indirect relationship between the two representations, via hypergraphs. We investigate more direct relationships between bid graphs and item graphs in Chapter 4.

We note here that Gottlob and Greco use a notion of combinatorial auction equivalence in order to prove their relationship between the hypergraph representations of CAs and valid item graphs. In order to guarantee that if $i \in I$, then $\{i\} \in B$, which was required above, we must allow certain bids of value 0. Such bids do not affect any solution to the winner determination problem and thus one can argue that the resulting combinatorial auction is equivalent to the original. Sandholm first used this method to prove optimality of an algorithm for solving WDP while maintaining an equivalent combinatorial auction [45]. Gottlob and Greco also used this definition of combinatorial auction equivalence to prove the result mentioned above [18]. While this definition has been useful, it does not help reduce an item-graph with treewidth two into an item-graph with treewidth one. We introduce another definition of combinatorial auction equivalence in Chapter 4, which gives us the ability to construct item graphs of smaller treewidth.

Chapter 3

Bid Graphs

In this chapter, we discuss some parameterizations of the winner determination problem for combinatorial auctions. We analyze each parameterized problem's hardness by considering the bid graph representation of its combinatorial auction. Recall the definition of the winner determination problem and combinatorial auctions from Section 2.3.1, and the bid graph construction from Section 2.3.3. The winner determination problem for a combinatorial auction is equivalent to finding a maximum weighted independent set on the bid graph of the auction [46, 48], where we define the weight of any vertex representing a bundle of items $S \subseteq I$ to be $B^*(S)$, as defined in Section 2.3.1.

In order to parameterize the winner determination problem we need to consider what factors may affect the running time of finding a solution. Given the simple construction and nature of WDP, we need to restrict either the number of vertices, the number of edges, or enforce some structure on the graph itself. There are many ways to accomplish this but not all restrictions lead to fixed-parameter tractable algorithms. We can restrict

any combination of the problem's parameters. The number of items, number of agents, number of atomic bids, and number of interactions between agents are some of the possible parameters. In this chapter, we consider a few of these parameters and also look at restricting the graph class of our bid graph.

In Section 3.1, we introduce a simple parameterization of the winner determination problem for combinatorial auctions, which we will denote as WD(k), where k is the parameter and is a positive integer. We prove that WD(k) is $W[1]$-complete, and thus fixed-parameter intractable. We then introduce another parameterization of WDP for CAs. We cite a dynamic programming algorithm by Rothkopf *et al.* that proves that this other parameterization is fixed-parameter tractable [44].

The rest of this chapter is split into two sections. In Section 3.2, we provide two parameterizations of WDP for CAs by parameterizing the variables of the auction itself. In Section 3.3, we investigate restricting the bid graph using properties of graphs. For example, we consider restricting the graph class of the bid graph, and combining this restriction with the simple parameterized problem WD(k), the details of which we leave to Section 3.3.

3.1 A Simple Parameterization of WDP for CAs

In this section, we introduce a simple parameterization of the winner determination problem for combinatorial auctions and show a proof that it is $W[1]$-complete, which can be attributed to a result by Downey and Fellows [11]; we review a proof of its $W[1]$-completeness for illustration of the use of parameterized complexity. We then introduce another param-

eterization of WDP for CAs and cite a dynamic programming algorithm by Rothkopf *et al.* that proves that it is fixed-parameter tractable [44]. Therefore, we note that it is an interesting problem to find other parameterizations of WDP for CAs, and prove whether such parameterizations are fixed-parameter tractable or fixed-parameter intractable.

We recall from Section 2.3.1 that a valid outcome is a collection $ValO = \{S_1, \ldots, S_\ell\}$, where $S_i \subseteq I$ for all $1 \leq i \leq \ell$, and $S_i \cap S_j = \emptyset$ for all $1 \leq i < j \leq \ell$. As previously stated in Section 2.3.1, a partial solution to the winner determination problem for a combinatorial auction is a valid outcome that maximizes, over all possible valid outcomes, the summation of bid values on the bundles of items contained in that outcome. Therefore, a partial solution to WDP for a CA is a collection of mutually disjoint atomic bids $\{(S_1, p_1), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell} p_j$ is maximal over all possible collections of mutually disjoint atomic bids. We recall from Section 2.3.1 that we are only interested in atomic bids (S_i, p_i) where $p_i = B^*(S_i)$, and that any items that are not included in a valid outcome we consider to be in set S' with a bid of zero; these items are kept by the seller at the end of the auction. Finally, we recall that for a given CA, the valid outcome of a partial solution to WDP together with S' is a solution to WDP for the auction. Therefore, we define a parameterized version of WDP as follows:

k-WINNER DETERMINATION (WD(k))

Input: A set of agents A, items I, and bids $B = B_1, B_2, \ldots, B_{|A|}$.

Parameter: Positive integer k.

Question: Does there exist a collection of mutually disjoint atomic bids $\{(S_1, p_1), (S_2, p_2), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell} p_j \geq k$?

From what we noted above, a solution to WD(k) is equivalent to a valid outcome

$ValO = \{S_1, \ldots, S_\ell\}$, such that $\sum_{j=1}^{\ell} B^*(S_j) \geq k$, since $p_j = B^*(S_j)$ for all $1 \leq j \leq \ell$. To show that WD(k) is $W[1]$-complete, and hence fixed-parameter intractable, we reduce from the following problem, which is known to be $W[1]$-complete [11]:

k-INDEPENDENT SET (IS(k))

Input:	A graph $G = (V, E)$.
Parameter:	Positive integer k.
Question:	Does there exist a set of vertices $V' \subseteq V$ of cardinality at least k such that for all $u, v \in V'$, $\{u, v\} \notin E$.

In order to be completely clear in how our parameters map from one parameterized problem to another in our fixed-parameter reductions, we denote one problem's parameter as k and the other problem's parameter as k'. This will be typical in all reductions that follow through the remainder of this thesis.

In Lemma 3.1, we show that WD(k) is $W[1]$-hard by reducing to WD(k) from IS(k'). This is a well-known result from the weighted set packing version of the problem, as discussed in Section 2.5, and is attributed to Downey and Fellows [11]. We provide details to the proof of Lemma 3.1 in order to illustrate the use of parameterized complexity theory in hardness proofs. We refer the reader to Section 2.4 where, following Definition 2.16, we describe how providing a fixed-parameter reduction from IS(k), a known $W[1]$-hard problem, to WD(k) can be used to show that WD(k) is also $W[1]$-hard. Our fixed-parameter reduction takes an instance of IS(k') and uses it to construct an instance of WD(k), where $k = k'$. To show that the reduction works, we need to prove that we have a solution to our constructed instance of WD(k) if and only if we have a solution to IS(k'). Further, to prove that the reduction is a fixed-parameter reduction we discuss its running time.

Lemma 3.1 ([11])**.** *WD(k) is W[1]-hard.*

[11]. We begin by providing a fixed-parameter reduction from IS(k') to WD(k). Given an arbitrary graph $G = (V, E)$ as input to IS(k'), we construct a combinatorial auction with $|V|$ agents. We number the edges of the graph as $1, 2 \ldots m$. For each vertex v, we create a new agent whose bid is a single atomic bid (S_v, p_v), where $S_v = \{\text{edge number of } e \in E \mid v \text{ an endpoint of } e\}$, and $p_v = 1$. Further, we set the parameter $k = k'$.

First, we assume that we have a collection of mutually disjoint atomic bids $\{(S_1, p_1), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell}(p_j) \geq k$. By construction, finding a collection of mutually disjoint atomic bids is equivalent to finding a set of mutually non-adjacent vertices, and thus equivalent to finding an independent set. Since for all atomic bids (S_v, p_v) we have $p_v = 1$, we also have $\ell \geq k$. Therefore, the number of mutually disjoint atomic bids in our collection is at least k. By construction, we have at least k mutually non-adjacent vertices. Therefore, we have an independent set of size at least $k' = k$.

We now assume that we have an independent set of size at least k'. By construction, we have at least k' mutually disjoint atomic bids $\{(S_1, p_1), \ldots, (S_\ell, p_\ell)\}$, one per vertex in the independent set, where $\ell \geq k'$. Also by construction, each atomic bid (S_j, p_j) contributes a value of $p_j = 1$ to the summation $\sum_{j=1}^{\ell} p_j$. Since $\ell \geq k'$, $\sum_{j=1}^{\ell} p_j \geq k = k'$.

The reduction is linear in the number of vertices and edges of G, with $k = k'$, and hence is a fixed-parameter reduction. Therefore, WD(k) is $W[1]$-hard. □

To prove that WD(k) is $W[1]$-complete, Lemma 3.2 shows that WD(k) is in $W[1]$. By Definition 2.18, WD(k) is in $W[1]$ if and only if WD(k) is fixed-parameter reducible to $L_{\mathcal{F}_{(1,h)}}$ for some h. Since IS(k') is $W[1]$-complete, we know that it is in $W[1]$. Hence, IS(k')

is fixed-parameter reducible to $L_{\mathcal{F}_{(1,h)}}$ for some h. Therefore, to prove that WD(k) is in $W[1]$, it suffices to show that WD(k) is fixed-parameter reducible to IS(k'), as is done in Lemma 3.2. Although this is a known result by Downey and Fellows, we provide a proof for illustration [11].

For our reduction in Lemma 3.2, we modify a bid graph G_B that has been constructed for a combinatorial auction in polynomial time. We recall from Section 2.3.3 that each vertex in G_B represents a distinct atomic bid (S_v, p_v), and that the weight of the vertex is $p_v = B^*(S_v)$. The agent associated with a vertex is not important for our reduction. In Lemma 3.2, for each vertex v in G_B, we create a group $\mathcal{V'}_v$ of pairwise non-adjacent vertices. All vertices in group $\mathcal{V'}_v$ are adjacent to all vertices in another group $\mathcal{V'}_u$, $u \neq v$, if and only if v was adjacent to u in the bid graph G_B. The number of vertices in each group is dictated by the minimum of our parameter k and the weight of v in the bid graph. Finally, we set the parameter k' of IS(k') to be k.

As was the case with Lemma 3.1, we show the reduction works by proving that we have a solution to our constructed instance of IS(k') if and only if we have a solution to WD(k). We conclude that the reduction is a fixed-parameter reduction by discussing its running time.

Lemma 3.2 ([11])**.** *WD(k) is in* $W[1]$.

[11]. We prove that WD(k) is in $W[1]$ by reducing the problem to IS(k'). In Section 2.3.3, we saw that we can construct a bid graph $G_B = (V_B, E_B)$ in polynomial time for a combinatorial auction. Each vertex $v \in V_B$, v represents exactly one distinct atomic bid (S_v, p_v). Further, for each distinct atomic bid (S_u, p_u) there exists exactly one vertex $u \in V_B$ that represents (S_u, p_u). To begin, we let $G' = (V', E')$ be the same graph as G_B,

with $V' = V_B$ and $E' = E_B$. To construct an instance of IS(k') we cycle through each vertex $v \in V_B$, and add $n = \min\{k, p_v\} - 1$ pairwise non-adjacent vertices $v_1, \ldots, v_n$ to G'. We let $\mathcal{V'}_v = \{v, v_1, \ldots, v_n\}$. A vertex that was added will be made adjacent to a vertex $u \in V'$ if and only if $\{v, u\} \in E'$. Essentially, every vertex v in G_B is now a set of $\min\{k, p_v\}$ pairwise non-adjacent vertices in G' that are all adjacent to exactly the same vertices in G'. We denote the completely constructed graph as $G = (V, E)$. Further, we set the parameter $k' = k$.

We now show that we have a solution to our constructed instance of IS(k') if and only if this solution maps, using our above construction technique, directly to a solution to WD(k), where $k = k'$. This is one of the requirements of a fixed-parameter reduction. First, we show that if we have a solution to WD(k), then this solutions maps, using our above construction technique, directly to a solution to IS(k'), where $k' = k$. Afterwards, we show that if we have a solution to some constructed instance of IS(k'), then we can map this solution to a solution to WD(k).

We assume that we have a solution to WD(k), which is a collection of mutually disjoint atomic bids $\mathcal{C} = \{(S_1, p_1), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell}(p_j) \geq k$. We prove that we can map this solution to a solution to our constructed instance of IS(k'). We let (S_j, p_j) be any one of the atomic bids in $\mathcal{C}$. Atomic bid (S_j, p_j) is represented by exactly one vertex $v_j \in V_B$. In G, there are $\min\{k, p_j\}$ mutually non-adjacent vertices associated with v_j, by construction; namely the vertices in set $\mathcal{V'}_{v_j}$. We consider any $(S_i, p_i) \in \mathcal{C}$, where $S_i \cap S_j = \emptyset$ (i.e. $i \neq j$). Atomic bid (S_i, p_i) is represented by exactly one vertex $v_i \in V_B$. By construction, there are $\min\{k, p_i\}$ mutually non-adjacent vertices associated with v_i; namely those in set $\mathcal{V'}_{v_i}$. Because $S_i \cap S_j = \emptyset$, vertex v_i is not adjacent to vertex v_j in G_B.

Therefore, by construction, for all $v'_i \in \mathcal{V'}_{v_i}$, and for all $v'_j \in \mathcal{V'}_{v_j}$, v'_i is not adjacent to v'_j.

We now show that the maximum number of pairwise non-adjacent vertices in G is at least $k' = k$. We let $\mathcal{V'}$ denote the set of vertices $\bigcup_{i=1}^{\ell} \mathcal{V'}_{v_i}$, where vertex v_i represents atomic bid $(S_i, p_i) \in \mathcal{C}$, for $1 \leq i \leq \ell$. From above, for all $u, v \in \mathcal{V'}$, $u \neq v$, we know that u is not adjacent to v in G. Thus, $\mathcal{V'}$ is a set of pairwise non-adjacent vertices. Therefore, the maximum number of pairwise non-adjacent vertices in G is at least the size of $\mathcal{V'}$. By definition, $|\mathcal{V'}| \geq \sum_{i=1}^{\ell}(\min\{k, p_i\})$. If for any i, $1 \leq i \leq \ell$, $\min\{k, p_i\} = k$, then $|\mathcal{V'}| \geq k$. If for all i, $\min\{k, p_i\} = p_i$, then since we assumed that $\sum_{j=1}^{\ell}(p_j) \geq k$ we also have $|\mathcal{V'}| \geq k$. Therefore, the number of pairwise non-adjacent vertices in G is at least $k' = k$. Thus, we have a solution to our constructed instance of IS(k'), where $k' = k$.

We have completed half of the proof of the statement that we have a solution to our constructed instance of IS(k') if and only if this solution maps, using our above construction technique, directly to a solution to WD(k), where $k = k'$. To complete the proof, we must show that if we have a solution to some constructed instance of IS(k'), then we can map this solution to a solution to WD(k). Thus, we assume that we have an independent set IS of G of size at least k'. Each vertex $v \in IS$ corresponds to a vertex u_v in G_B, and vertex u_v corresponds to an atomic bid (S_{u_v}, p_{u_v}). Therefore, indirectly, each vertex $v \in IS$ represents an atomic bid. We let the number of distinct atomic bids represented by all of the vertices in IS be $\ell' \leq k'$, and we denote the distinct atomic bids by $(S_{i_1}, p_{i_1}), \ldots, (S_{i_{\ell'}}, p_{i_{\ell'}})$. By construction, $\sum_{j=1}^{\ell'} p_{i_j} \geq k'$. Specifically, the number of vertices in IS associated with an atomic bid (S_{i_j}, p_{i_j}) is at most p_{i_j}. Therefore, since the number of vertices in IS is at least k', $\sum_{j=1}^{\ell'} p_{i_j} \geq k = k'$. Thus, we have a collection of mutually disjoint atomic bids $\{(S_{i_1}, p_{i_1}), \ldots, (S_{i_{\ell'}}, p_{i_{\ell'}})\}$ such that $\sum_{j=1}^{\ell'} p_{i_j} \geq k$.

We now argue that the reduction is a fixed-parameter reduction by showing that the reduction requires time that is fixed-parameter tractable, as required by Definition 2.16. First, the number of vertices in the constructed graph G is at most $|V_B| \cdot k$. If we use an adjacency-list representation for the graph, then each vertex corresponds to an unsorted list of vertices to which it is adjacent. Adding an edge between two vertices can be done in constant time; that is, given two vertices i and j, we can add i to j's list, and vice-versa, in constant time. Adding a new vertex also requires only constant time. We assume that we use an adjacency-list representation for our graphs. The addition of each vertex to G' requires time at most $O(|V_B| \cdot k)$. In more detail, for each vertex $v \in V_B$, we add at most $k-1$ vertices to G'. For each of these vertices that we add, we must cycle through at most $|V_B| \cdot k$ other vertices in G' using v's adjacency list. For each vertex u that is adjacent to v in G', we make the added vertex adjacent to u as well. For each added vertex we require time at most $O(|V_B| \cdot k)$ to add the necessary edges to G'. Therefore, for each vertex $v \in V_B$, we require time at most $O(|V_B| \cdot k^2)$ to add up to $k-1$ vertices to G', along with the necessary edges. Therefore, if we use the adjacency-list representation for our graphs, our reduction requires time at most $O(|V_B|^2 \cdot k^2)$, which is fixed-parameter tractable.

Therefore, since $k' = k$ in our reduction, WD(k) is in $W[1]$. □

By Lemma 3.1 and Lemma 3.2, WD(k) is $W[1]$-complete. Also, as a consequence of the proofs of the lemmas, we have shown the equivalence in both directions of WD(k) and IS(k). Thus, we can think of an instance of the winner determination problem in terms of the independent set problem on a graph.

Theorem 3.3 ([11])**.** *WD(k) is $W[1]$-complete.*

The results of Lemma 3.1 and Theorem 3.3 imply that we cannot find a fixed-parameter tractable algorithm to WD(k), unless FPT = $W[1]$ [11]. We have demonstrated that WDP for CAs is computationally "harder" than some parameterizations of known NP-complete problems. For example, Downey and Fellows show a parameterization of vertex cover, a known NP-complete problem, that is fixed-parameter tractable [11].

To see that not all parameterizations of WDP for CAs are fixed-parameter intractable, we consider a dynamic programming algorithm that was introduced by Rothkopf *et al.* [44]. We recall from Section 2.3.1 that $B^*(S) = \max\{B_i(S) \mid B_i(S) \text{ submitted }\}$. The algorithm by Rothkopf *et al.* considers $B^*(S)$ for every $S \subseteq I$, in order to determine a solution to WDP. The maximal bid value for each bundle of items S is either $B^*(S)$, or it is the sum of the maximal bid values of two disjoint bundles of items S_1 and S_2 such that $S_1 \cup S_2 = S$. For any bundle of items $S \subseteq I$, we call disjoint bundles of items S_1, S_2 such that $S_1 \cup S_2 = S$, *two disjoint exhaustive subsets of* S. Rothkopf *et al.* proceed to calculate the maximal bid values for each $S \subseteq I$ by beginning with the smallest possible S and increasing in size until $S = I$. They define set $C(S)$ as follows: $C(S) = S$ if $B^*(S)$ is greater than the sum of maximal bid values of any two disjoint exhaustive subsets of S; otherwise, $C(S)$ is the smaller of the two disjoint exhaustive subsets of S resulting in a maximal bid value summation, when compared with any other two disjoint exhaustive subsets of S. Finally, starting with an exhaustive valid outcome of $ValO = \{I\}$, Rothkopf *et al.* iteratively use their definition of $C(S)$ for each $S \in ValO$, until for every $S \in ValO$, $C(S) = S$. If there exists $S \in ValO$ such that $C(S) \neq S$, they modify $ValO$ by removing S and adding the sets $C(S)$ and $S \setminus C(S)$. Their algorithm, if implemented correctly, has a running time of $o(2^{2|I|})$ and $\Omega(2^{|I|})$ [44].

We construct a new parameterization of WDP for CAs by considering the definition of the parameterized problem WD(k). We modify the definition by adding the parameter $MaxItems$, requiring that $|I| \leq MaxItems$ and asking the following question: Does there exist a collection of mutually disjoint atomic bids $\{(S_1, p_1), (S_2, p_2), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell} p_j \geq k$? We denote this new parameterized problem by WD(MI, k) and we now show how it is trivially fixed-parameter tractable. We do this in order to demonstrate that an obvious parameterization of WDP is fixed-parameter tractable. From above, if $|I| \leq MaxItems$, then there exists a dynamic programming algorithm that finds a solution to WDP for CAs with this restriction in time at most $o(2^{2 \cdot MaxItems})$ [44]. Since $MaxItems$ is a parameter of WD(MI, k), the algorithm is exponential in the parameter $MaxItems$ but constant in the remainder of the input of WD(MI, k), and thus is potentially a fixed-parameter tractable solution to WD(MI, k). The dynamic programming algorithm by Rothkopf *et al.* returns the optimal solution to WDP for CAs. Therefore, we have a fixed-parameter tractable algorithm that returns the optimal solution to WDP for CAs where $|I| \leq MaxItems$. If we have the optimal solution $\{(S_1, p_1), (S_2, p_2), \ldots, (S_\ell, p_\ell)\}$, we can simply verify that $\sum_{j=1}^{\ell} p_j \geq k$. Therefore, the dynamic programming algorithm above by Rothkopf *et al.* is a fixed-parameter tractable algorithm for WD(MI, k).

We have now seen a fixed-parameter tractable parameterization of combinatorial auctions; namely, WD(MI, k). It is clear that different parameterizations of WDP lead to different parameterized complexity classifications; WD(k) is $W[1]$-complete, but WD(MI, k) is fixed-parameter tractable. Therefore, we note that it may be of interest to find other parameterizations of WDP for CAs, and prove whether such parameterizations are fixed-parameter tractable or fixed-parameter intractable. To follow, in Sections 3.2 and 3.3, we

investigate other parameterizations; some of which are fixed-parameter tractable and some of which are not.

3.2 Parameterizing the Auction

We will now demonstrate some of the ways one can parameterize WDP and the effect, if any, each parameterization has on its complexity when compared with WD(k), as introduced in Section 3.1. There are many choices to be made and the ones we make in this section are by no means exhaustive; there may be many other parameterizations for which the problem remains hard when compared with WD(k), and others that are fixed-parameter tractable. We begin by parameterizing the number of distinct atomic bids in a CA.

The number of vertices in the bid graph, defined in Section 2.3.3, is exactly the number of distinct atomic bids across all agents, $dab(B)$. A simplistic algorithm would search through every possible combination of vertices to try and find the maximum weighted independent set. The number of edges is bounded above by $\binom{dab(B)}{2}$, and so such a naive algorithm would require time of at most $O(dab(B)^2 \cdot 2^{dab(B)})$. What is interesting here is that the number of items does not affect the result at all. We recall that our algorithm for constructing the bid graph takes time $O(M|I| + dab(B)|I| + |E| \cdot |I|)$ time to complete, where $M = \sum_{i=1}^{|A|} |B_i|$ and $|E|$ is the total number of edges in the completed bid graph. Then, the total required time to find a solution to WDP is $O(M|I| + dab(B)|I| + |E| \cdot |I| + dab(B)^2 \cdot 2^{dab(B)})$. This leads to the following parameterization:

DAB, k-**WINNER DETERMINATION (WD(DAB, k))**

Input: Agents A, items I, and bids $B = B_1, B_2, \ldots, B_{|A|}$, where $dab(B) \leq DAB$

Parameter: Positive integers DAB, k.

Question: Does there exist a collection of mutually disjoint atomic bids $\{(S_{i_1}, p_{i_1}), (S_{i_2}, p_{i_2}), \ldots, (S_{i_\ell}, p_{i_\ell})\}$ such that $\sum_{j=1}^{\ell}(p_{i_j}) \geq k$?

The above algorithm constructs a bid graph in time that is polynomial in the number of items and agents, but finds a maximum weighted independent set of the bid graph in time that is exponential in the number of distinct atomic bids. Since the number of distinct atomic bids is bounded by our parameter DAB, we have a fixed-parameter tractable algorithm. Therefore, WD(DAB, k) is fixed-parameter tractable, since we can find the maximum weighted independent set and then verify that its value is at least k.

A practical problem arises from this parameterization as the seller has no direct control over the total number of atomic bids. However, the seller knows the total number of atomic bids before beginning to solve WDP, and therefore may check that the number of distinct atomic bids is below some threshold, and if so run an algorithm that finds a maximum weighted independent set. If the number of distinct atomic bids is not below the threshold, then the seller may opt to run a different algorithm, which may depend on other properties of the auction.

Without additional restrictions on the number of agents, $|A|$, the seller cannot specify a restriction per agent based on knowledge of a bound on $dab(B)$. We consider another perspective: parameterize the number of distinct atomic bids per agent i as $DABi$. We note that $DABi$ is assumed to be the same for all agents. If each agent had a different

parameter that bounded their allowed number of distinct atomic bids, then setting $DABi$ to be the maximum parameter value over all agents allows our results that follow to hold. By parameterizing the number of distinct atomic bids per agent as $DABi$, the value of $dab(B)$ is at most $|A| \cdot DABi$, and thus the required running time for the naive algorithm mentioned above becomes $O(|A|^3 \cdot DABi^2 \cdot |I|^2 + (|A| \cdot DABi)^2 \cdot 2^{|A| \cdot DABi})$. The naive algorithm is now exponential in $|A|$, which is not a parameter. However, we cannot conclude that this parameterized problem is fixed-parameter intractable without a proof. First, we formally define the parameterized problem:

$DABi, k$-**WINNER DETERMINATION (WD($DABi, k$))**

Input: Agents A, items I, and bids $B = B_1, B_2, \ldots, B_{|A|}$, where the number of distinct atomic bids in any B_i is at most $DABi$.

Parameter: Positive integers $DABi, k$.

Question: Does there exist a collection of mutually disjoint atomic bids $\{(S_{i_1}, p_{i_1}), (S_{i_2}, p_{i_2}), \ldots, (S_{i_\ell}, p_{i_\ell})\}$ such that $\sum_{j=1}^{\ell}(p_{i_j}) \geq k$?

Lemma 3.4. *WD($DABi, k$) is $W[1]$-hard.*

Proof. We perform a fixed-parameter reduction from WD(k') to WD($DABi, k$). Given an instance of WD(k') with agents A', items I', and bids $B' = B'_1, \ldots, B'_{|A'|}$, we construct an instance of WD($DABi, k$) where the number of agents in WD($DABi, k$), $|A|$, is $dab(B')$. Each agent is assigned exactly one distinct atomic bid from WD(k'). The item set, I, will be the same as I', and as a consequence of $|A| = dab(B')$, the bids in our instance of WD($DABi, k$) will be $B = B_1, \ldots, B_{dab(B')}$. We assume that each agent only submits one bid; namely, the bid they were assigned from the instance of WD(k'). We have constructed

an instance of WD($DABi, k$), where $DABi = 1$ and $k = k'$. The reduction is polynomial in the size of the bid graph of WD(k'), which we proved in Section 2.3.3 is polynomial in the size of the input to WD(k'). Since the bid graphs of both WD(k') and WD($DABi, k$) are the same, a maximum weighted independent set of either graph has the same weight. Therefore, we have a collection of mutually disjoint atomic bids $\{(S_{i_1}, p_{i_1}), (S_{i_2}, p_{i_2}), \ldots, (S_{i_\ell}, p_{i_\ell})\}$ in WD($DABi, k$) if and only if $\{(S_{i_1}, p_{i_1}), (S_{i_2}, p_{i_2}), \ldots, (S_{i_\ell}, p_{i_\ell})\}$ is also a collection of mutually disjoint atomic bids in WD(k'); the total weight of any such collection is the same in both problem instances. Thus, WD($DABi, k$) is $W[1]$-hard. □

It follows from Lemma 3.4 that parameterizing by the number of distinct atomic bids per agent is not worthwhile, as the problem has the same parameterized hardness as WD(k). However, if we parameterize the number of distinct atomic bids per agent and the number of agents, then we have a fixed-parameter tractable parameterized problem because even our naive algorithm was exponential only in $|A|$ and $DABi$. In real-world applications, enforcing this parameterization is likely more feasible than parameterizing the total number of distinct atomic bids over all agents.

An often studied special case of combinatorial auctions restricts agents to bidding on at most a single bundle of items [34, 39, 33, 2]. In the literature, agents are referred to as *single-minded* when faced with this restriction [34, 33]. Due to the relative importance of this special case, we should note that Lemma 3.4 gives us a proof of the hardness when agents are single-minded. In fact, the case is equivalent to forcing the value of $DABi$ to 1.

Lemma 3.4 states that WD($DABi, k$) is $W[1]$-hard, but more importantly the proof is exactly the same for the case where agents are single-minded. Therefore, WDP for a CA that restricts agents to bidding in a single-minded manner is also $W[1]$-hard. For a further

and more in-depth discussion of this special case, we refer the reader to Lehmann *et al.* [33].

3.3 Restricting the Graph Class

As an additional avenue of investigation, we consider what happens if we restrict agents to certain types of bids. We define *agent i's bid graph* as the bid graph obtained from the construction algorithm in Section 2.3.3 while only considering bids submitted by agent i. To restrict agents to certain types of bids, we restrict each agent to bidding such that the agent's bid graph maintains a desired structure. The question we then ask is how this structure affects the hardness of WDP for the overall combinatorial auction; is WDP for the auction fixed-parameter tractable, or is it $W[1]$-hard like WD(k)? It is important that we note that while we are restricting the agents' bids in order to extract proof of fixed-parameter tractability, we are not necessarily proposing that the agents be restricted in practice; rather, we are investigating those structures that are useful and those that are not useful. We comment further on this distinction at the end of this section.

First, we present a general problem definition, in which we restrict our problem using some desired graph class and parameterize by k. We denote the desired graph class by β, and we require that each agent's bid graph belong to graph class β. The parameter k will be used in the typical way we have seen previously, such as in WD(k). After investigating the winner determination problem for a few specific graph classes, we form a more general theorem for numerous graph classes. Finally, at the end of this section, we consider a bid graph where the number of vertices in each connected component is bounded from above.

β, k-WINNER DETERMINATION (WD(β, k))

Input:	A set of agents A, items I, and bids $B = B_1, B_2, \ldots, B_{\|A\|}$, where agent i's bid graph belongs to graph class β.
Graph Class:	β.
Parameter:	Positive integer k.
Question:	Does there exist a collection of mutually disjoint atomic bids $\{(S_{i_1}, p_{i_1}), (S_{i_2}, p_{i_2}), \ldots, (S_{i_\ell}, p_{i_\ell})\}$ such that $\sum_{j=1}^{\ell}(p_{i_j}) \geq k$?

Analyzing the general problem is of course very difficult. We cannot immediately generalize all graph classes and give one result. Instead, we first start with a couple of lemmas before building up to a more general result. Intuitively, we will show in Theorem 3.7 that if β does not restrict the interaction between the atomic bids of different agents, then the resulting WD(β, k) problem is $W[1]$-hard.

Observation 3.5. *For β the class of interval graphs defined by the interval $\mathcal{I} = (1, 2, \ldots, |I|)$, WD($\beta, k$) is fixed-parameter tractable. Specifically, parameter k is not even necessary, as finding the optimal solution to WDP with β as defined above is tractable.*

As stated in Observation 3.5, all agents use the interval $\mathcal{I} = (1, 2, \ldots, |I|)$ and as such each atomic bid in B_i must consist of a subinterval of $\mathcal{I}$, for each agent i. Further, consider the bid graph of the complete combinatorial auction consisting of all atomic bids. As an extension our first statement, each vertex of the CAs bid graph represents a bid consisting of a subinterval of $\mathcal{I}$. Therefore, the bid graph of the resulting CA is by definition an interval graph. We refer the reader back to Section 2.1 for the complete and formal definition of an interval graph. Interval graphs are chordal graphs [17], and hence the maximum

weighted independent set problem on interval graphs has an algorithm that runs in linear time [13]. The winner determination problem for combinatorial auctions is equivalent to finding the maximum weighted independent set of the bid graph, and thus we have a linear time algorithm for WD(β, k).

From Observation 3.5, we see that if we restrict each agent's bid graph to interval graphs of one common interval, then the bid graph of the resulting combinatorial auction is also an interval graph. Therefore, the winner determination problem can be solved in linear time given these restrictions. Rothkopf *et al.* discussed a combinatorial auction where the items can be ordered and it is required that bids be placed on consecutive items [44]. While the paper presented did not make use of interval graphs directly, Rothkopf *et al.* were actually specifying that each agent's bid graph be an interval graph, where the interval was the same for all agents.

It is important that we note that it is more important that we use an interval common to all agents than it is we use interval graphs in particular. This global property that is shared among all agents is key. For example, chordal graphs can be defined as intersection graphs of subtrees of a tree [16]. Thus, if all agents are forced to make their bid graph intersection graphs of subtrees of a common tree, then the bid graph for the auction including all bids from all agents will be chordal. Thus, Observation 3.5 also applies to chordal graphs, so long as all agents use the same global tree when constructing their chordal graph. A natural question is to ask what happens if we remove such a global constraint. For example, let's say that β is the class of graphs such that a single vertex is a member of graph class β; in this case, we say that *graph class β admits singletons.*

Lemma 3.6. *For β defined as a graph class that admits singletons, the WD(β, k) problem*

is $W[1]$-hard.

Proof. In order to show that WD(β, k) is $W[1]$-hard for β defined above, we give a fixed-parameter reduction from WD(k') to WD(β, k). Given an arbitrary instance of WD(k') and its bid graph $G' = (V', E')$, we construct a new CA where each agent's bid graph is a single vertex, which by definition is a member of graph class β. We let the number of agents in our new CA, $|A|$, be the same as the number of vertices of G', and assign exactly one vertex in G' to a distinct agent. We let agent i's bid in the new auction be the same as the atomic bid associated with the vertex in G' assigned to agent i. Therefore, every agent's bid graph of one vertex belongs to graph class β. We have constructed a new combinatorial auction with $|A| = |V|$ agents, each of which places exactly one atomic bid, and where each agent's bid graph is a member of β. This reduction requires time polynomial in the size of the original auction, and finally we note that $k = k'$.

Since the bid graphs of the two auctions are identical in every way, WD(β, k) is $W[1]$-hard, where β admits singletons. □

Given that we cannot restrict β to this class of graphs, a natural question is to ask for which graph classes is WD(β, k) $W[1]$-hard. While we do not provide a complete characterization of all of possible graph classes resulting in $W[1]$-hardness, we do provide a partial characterization.

We delve further into the details about WD(β, k) for cases where β does not admit a singleton, because in some cases a seller may be interested in knowing at what point restricting the agents of their auction is going to help. Thus, we will spend a bit more time answering this question for a more generic class of graphs.

First, we define what it is to be a minimal β graph. We choose to quantify the *size of a graph* by the sum of the number of vertices and edges in the graph. A *non-empty graph* is a graph with at least one vertex. A *minimal β graph* is a non-empty graph contained in graph class β whose size is at most the size of any other non-empty graph from graph class β. Intuitively, it is a non-empty graph from graph class β whose size is minimal. The size of the input to WD(β, k) is measured as $|A| + |I| + \sum_{i=1}^{|A|} sizeof(B_i)$, where $sizeof(B_i)$, the size of the bid $B_i = \{(S_1, p_1), \ldots, (S_{j_i}, p_{j_i})\}$ submitted by agent i, is defined to be $sizeof(B_i) = \sum_{k=1}^{j_i} |S_k|$.

It is possible for a minimal β graph to have infinite size, or to have size that is exponential in the input to WD(β, k), and so our first restriction in Theorem 3.7 is that β have a minimal β graph of size at most a polynomial in the size of the input to WD(β, k). A minimal β graph will change in size only if the definition of class β changes. Therefore, if the size of a minimal β graph is a function of the input to WD(β, k), the definition of graph class β must also be some function of the input to WD(β, k). In practical applications, the size of the input to WD(β, k) would be finite. The restriction to the size of a minimal β graph is necessary because we need our reduction in the proof of the theorem to be a fixed-parameter reduction. Further, we require that β imposes no restriction on the interaction between the atomic bids of different agents. In Theorem 3.7, we show that under these restrictions, the resulting WD(β, k) problem is $W[1]$-hard.

From what we just mentioned, the graph class β may be defined by some function of the input to WD(β, k), as long as the size of a minimal β graph is at most a polynomial in the size of the same input. If graph class β is a function of the input to WD(β, k) then the class definition can potentially change throughout the bidding process. We assume that

the graph class β is not a function of any property of the bids submitted by the agents; this assumption is reasonable because if we restrict all agents' bid graphs to class β while β is changing throughout the bidding process, then each agent will be restricted differently. Such an auction will not be incentive compatible, because it is a better strategy for an agent to wait until all other agents have bid before placing their own bid, in order to allow themselves more flexibility in how they are allowed to bid. This could lead to no agent placing a bid, because every agent is waiting. As a consequence of this assumption, the size of a minimal β graph cannot be a function of any property of the bids submitted by the agents. More generally, for the same reason, we assume that the definition of graph class β is not a function of any part of the input to WD(β, k) that is unknown before the start of the auction.

The idea behind the proof of Theorem 3.7 is that we replace each vertex in the bid graph of an instance of WD(k') with a copy of a minimal β graph. We fix our attention on one specific minimal β graph, which we denote by G_β, and we let IS_β denote a maximum independent set of G_β. All vertices in the copy of G_β that replace a vertex have weight equal to that of the weight of the vertex they replace. We let G' be the bid graph of an instance of WD(k'), and denote our constructed bid graph as G. We prove in Theorem 3.7 that there exists an independent set of G' of weight at least k' if and only if there exists an independent set of G of weight at least $k = |IS_\beta| \cdot k'$.

Theorem 3.7. *The definition of graph class β is not a function of any part of the input to WD(β, k) that is unknown before the start of the combinatorial auction. If β has a minimal β graph of size at most a polynomial in the size of the input to WD(β, k), which imposes no restriction on the interaction between the atomic bids of different agents, then*

the resulting WD(β, k) problem is W[1]-hard. A minimal β graph representation is given as $G_\beta = (V_\beta, E_\beta)$, and does not need to be discovered.

Proof. To show that WD(β, k) is $W[1]$-hard, we reduce from WD(k'). Given an instance of WD(k'), we consider its bid graph $G' = (V', E')$. We aim to construct a bid graph $G = (V, E)$ that represents an instance of the WD(β, k) problem. For this, we consider each vertex in G' to be a distinct agent for our new problem. We assign each vertex in G' to exactly one distinct agent of WD(β, k). We now have a single atomic bid per agent in WD(β, k) (and vice-versa), but an agent's bid graph is not necessarily a β graph. We let G be a copy of G'. We note that currently, G is the bid graph of our new constructed combinatorial auction.

To make each agent's bid graph a β graph, we replace each vertex in G with a copy of G_β. We assign each vertex in the copy of G_β to the same agent as the original vertex in G. Further, for each vertex in the copy of G_β, we assign the vertex a weight equal to the weight of the original vertex in G that the copy replaced. Each agent in our new problem is now assigned to exactly one minimal β graph in G (and vice-versa), and exactly one distinct vertex in G'. For a vertex v in G', we refer to the agent assigned to vertex v as agent v. We refer to the minimal β graph in G that replaced v in G' as agent v's minimal β graph. If two vertices v_1, v_2 were adjacent in G', then every vertex in agent v_1's minimal β graph must be made adjacent to every vertex in agent v_2's minimal β graph in G; we describe this as agent v_1's minimal β graph being adjacent to agent v_2's minimal β graph. Finally, we let $k = |IS_\beta| \cdot k'$.

As we stated in Section 2.3.3, the algorithm for constructing a bid graph also provides a method for converting a bid graph into a combinatorial auction that runs in polynomial

time. Therefore, we translate our constructed bid graph to be our new combinatorial auction. Each agent's bid graph is now a minimal β graph, by construction.

Bid graph G then represents a new combinatorial auction where the number of agents is equal to the number of vertices of the original graph G', and each agent's bid graph is a copy of the minimal β graph G_β, where each vertex in any single copy of G_β in G has the same weight. We denote a maximum independent set of G_β as IS_β. Since the vertices' weights are all equal in each agent's bid graph, which is a minimal β graph, $|IS_\beta|$ must be the same as the number of vertices in any maximum weighted independent set of agent v's minimal β graph.

First, we show that if G' has an independent set $IS_{G'}$ of weight at least k', then G has an independent set of weight at least $k = |IS_\beta| \cdot k'$. Thus, we assume that G' has an independent set $IS_{G'}$ of weight at least k' and denote the vertices of $IS_{G'}$ as $v_1, v_2, \ldots, v_\ell$. Each vertex v_i in bid graph G' represents a distinct atomic bid (S_i, p_i), by construction, and the weight of vertex v_i is p_i. Therefore, $\sum_{i=1}^{\ell} p_i \geq k'$. The vertices of $IS_{G'}$ map to ℓ distinct agents in our new CA, each with a bid graph that is a copy of G_β. Originally, each vertex $v_i \in IS_{G'}$ contributed weight p_i. For our new CA, we construct an independent set IS_G of G. For each $v_i \in IS_{G'}$, we add to IS_G the vertices in G associated with IS_β from agent v_i's minimal β graph. These maximum independent sets contribute a total weight of $\sum_{i=1}^{\ell} |IS_\beta| \cdot p_i \geq |IS_\beta| \cdot k'$, because $\sum_{i=1}^{\ell} p_i \geq k'$. By construction, IS_G is an independent set and therefore we have an independent set of our new bid graph of weight at least $k = |IS_\beta| \cdot k'$.

We now show that if G has an independent set IS_G of weight at least $k = |IS_\beta| \cdot k'$, then the original graph G' contains an independent set of weight at least k'. We consider exactly

one vertex in G from each copy of G_β that has a vertex in IS_G, and denote this set of vertices as $IS_{G'}$. As IS_β is maximal by definition, the vertices of $IS_{G'}$ must have a weight of at least k' from our definition that $k = |IS_\beta| \cdot k'$. Further, each vertex of $IS_{G'}$ maps directly to a distinct vertex in the original graph, G', and we denote this set of distinct vertices as $IS^*_{G'}$. Since the edge adjacencies of G' were preserved in our construction of G, the agents' minimal β graphs are adjacent to each other in G if and only if the agents' corresponding vertices were adjacent to each other in G'. Therefore, $IS^*_{G'}$ is an independent set of G' of weight at least k'.

Finally, we show that our reduction is fixed-parameter tractable. The size of the minimal β graph is at most a polynomial in the size of the input to WD(β, k), and is not a function of any property of the bids submitted by the agents. Therefore, the size of the minimal β graph is at most a polynomial in the size of the input to WD(k'). The construction of G is simply a multiplication of the size of the minimal β graph and the size of the original graph G'. Specifically, for each vertex v in G', we add new vertices and edges. The number of new vertices and edges is equal to the size of G_β, which is at most a polynomial in the size of the input to WD(k'). We denote the size of G_β by $|G_\beta|$. For each vertex added, we add at most $|V'| \cdot |V_\beta|$ edges. Thus, the construction requires time at most $O(|V'|(|G_\beta| + |V'| \cdot |V_\beta|^2)) \in O(|V'| \cdot |G_\beta| + |V'|^2 \cdot |V_\beta|^2)$. Because $|G_\beta|$, and thus $|V_\beta|$, is at most a polynomial of the size of the input to WD(k'), we have a fixed-parameter reduction from WD(k') to WD(β, k). Therefore, WD(β, k) is $W[1]$-hard. □

As a final note on the construction of G in our proof of Theorem 3.7, both stipulations in the theorem's statement are necessary. Our assumption in the beginning of the proof was that we were given an arbitrary instance of WD(k'), and hence for G to be a bid graph

of an instance of WD(β, k) we must allow arbitrary interactions between different agents' minimal β graphs. The other stipulation concerning the size of the minimal β graph was clearly used to show that the reduction was fixed-parameter tractable.

With Theorem 3.7, we have a partial characterization of WD(β, k) where we have outlined special cases for which the problem remains fixed-parameter intractable. We have also shown that when β is the class of interval graphs defined by the interval $\mathcal{I} = (1, 2, \ldots, |I|)$, then WD$(\beta, k)$ is fixed-parameter tractable; in fact it can be solved in polynomial time.

As a final example of a fixed-parameter reduction, we investigate the winner determination problem when the combinatorial auction's bid graph is a graph such that the number of vertices in each connected component is bounded from above. We recall from Section 2.1 that a *connected component* of a graph $G = (V, E)$ is the induced subgraph of a set of vertices V', such that the following holds: $V' \subseteq V$; the induced subgraph of G on V' is connected; and there does not exist any other V'' with $V' \subset V'' \subseteq V$ such that the induced subgraph of G on V'' is connected. When we refer to a component of a graph, we are referring to a connected component. We consider a combinatorial auction's bid graph, $G = (V, E)$, and we denote by $C_1, C_2, \ldots, C_q$ the connected components of G. We note that the number of vertices in the bid graph is not restricted; this number is still $dab(B)$.

By our construction in Section 2.3.3, every vertex of bid graph G represents a distinct atomic bid on a bundle of items, and hence there are at most $dab(B)$ components. Since there is at least one component, to solve WDP we must find a maximum weighted independent set of each of the components.

Lemma 3.8. *A maximum weighted independent set of a graph G has the same weight as the union of a maximum weighted independent set of each connected component of G. We*

let IS denote any particular maximum weighted independent set of G, and we let $W(IS)$ denote its weight. Further, we let IS_U denote the union of maximum weighted independent sets, one for each connected component of G, and we let $W(IS_U)$ denote the combined weight of all vertices in IS_U. Then $W(IS) = W(IS_U)$.

Proof. First we note that $W(IS) \geq W(IS_U)$. We assume that $W(IS_U) < W(IS)$ and derive a contradiction. IS and IS_U must differ in some of the vertices they use. We let IS_C denote all vertices in IS from any single component, C.

For $W(IS)$ to be maximal, IS_C must be a maximum weighted independent set of C. Otherwise, we contradict IS's maximality by choosing the maximum weighted independent set of C from IS_U; these vertices must differ from IS_C. By our assumption, at least one component C^* must exist such that $W(IS_{C^*})$ is greater than the weight of the maximum weighted independent set of C^* from IS_U. This is contradicts the maximality of our choice of a weighted independent set of C^* for IS_U. Therefore $W(IS) = W(IS_U)$. □

We now formulate a parameterized problem with a bid graph G where the number of vertices in each connected component of G is bounded from above.

C-Component ℓ, k-WINNER DETERMINATION (WD(C, ℓ, k))

Input:	A set of agents A, items I, and bids $B = B_1, B_2, \ldots, B_{\|A\|}$, where the bid graph has C components, each of which have at most ℓ vertices.
Graph Class:	C connected components, each of which have at most ℓ vertices.
Parameter:	Positive integers k, ℓ.

Question: Does there exist a set of mutually disjoint atomic bids $\{(S_{i_1}, p_{i_1}), (S_{i_2}, p_{i_2}), \ldots, (S_{i_\ell}, p_{i_\ell})\}$ such that $\sum_{j=1}^{\ell}(p_{i_j}) \geq k$?

We note that C is not a parameter, but rather a variable that is bounded from above by the number of distinct atomic bids, $dab(B)$.

Observation 3.9. *WD(C, ℓ, k) is fixed-parameter tractable.*

Proof. In Section 3.2, we introduced the parameterized problem WD(DAB, k). We break down the decision problem WD(DAB, k) into components. The number of vertices in each component is bounded from above by ℓ. Thus, a maximum weighted independent set of each component can be computed separately. Computing the maximum weighted independent set of a connected component requires a running time at most $O(\ell^2 \cdot 2^\ell)$, if implemented correctly. Lemma 3.8 states that we can combine the vertices from each component's maximum weighted independent set to form a maximum weighted independent set of the entire bid graph. We recall from Section 2.3.3 that the time required to construct a bid graph is $O(M|I| + dab(B)|I| + |E| \cdot |I|)$, where $M = \sum_{i=1}^{|A|} |B_i|$ and $|E|$ is the total number of edges in the completed bid graph. It follows that there exists a solution to WD(C, ℓ, k) requiring a running time of at most $O(M|I| + dab(B)|I| + |E| \cdot |I| + C \cdot \ell^2 \cdot 2^\ell)$. □

A problem arises in how this parameterization can be accomplished in practice. The seller must enforce a global restriction resulting in a bid graph of C connected components, each having a bounded number of vertices. There are at least two ways of accomplishing this. First, recognizing the structure of such a bid graph can be accomplished in polynomial time. In particular, the seller may construct the bid graph and then determine if it is a

graph of C connected components, each of which have at most ℓ vertices. If the graph follows the necessary restrictions, then we can solve the problem, otherwise we simply do not try.

As an alternate method, if it is practical, then the seller may separate items into groups of bounded size, and no agent may place items from different groups in the same atomic bid. By separating items into groups of size at most ℓ', we divide the items into $\lceil \frac{|I|}{\ell'} \rceil$ groups of size at most ℓ'. Agents must be instructed that none of their atomic bids may include items from more than one group. With this restriction we have bounded the number of possible bids per group to at most $2^{\ell'}$. By setting $\ell = 2^{\ell'}$, we have a partially constructed instance of WD(C, ℓ, k), where k needs to be set.

In the investigation of combinatorial auctions as they apply to specific economic areas, one may find that agents tend to bid such that their individual bid graphs belong to a particular graph class. If this is the case, it is natural for a seller to consider restricting all agents to bid in this manner in exchange for a faster solution to the winner determination problem. From our work in this section, we now have two examples where finding said particular graph class is useful information; namely interval graphs and bid graphs where the number of vertices in each connected component is bounded from above. We also have a few examples where finding the particular graph class is not useful. Should WD(β, k) be fixed-parameter intractable when restricted to β, it is useless for the seller to restrict agents to bid graphs belonging to graph class β because WD(k) with no such restriction has the same parameterized complexity.

Chapter 4

Item Graphs

While item graphs and bid graphs differ in how they represent combinatorial auctions, the underlying instance of the auction they represent is the same. As such, we wish to determine how bid graphs and item graphs relate to one another. We refer the reader to Chapter 2, Section 2.3.4 for the introduction to item graphs, and to Definition 2.13 for the formal definition of a valid item graph. In Section 4.1, we investigate classes of item graphs and whether they enforce enough structure on their combinatorial auctions to constrain the class of bid graphs associated with these auctions.

Turning our investigation beyond the direct relationship between item graphs and bid graphs, we discuss the possibility of modifying a combinatorial auction such that the bid graph of the new auction is equivalent to the bid graph of the original, unmodified auction. Equivalent combinatorial auctions are useful because a solution to the winner determination problem for one combinatorial auction can be easily translated into a solution to WDP for any equivalent combinatorial auction. In Section 4.2, we give our definition of bid graph

equivalence and show a method for modifying a combinatorial auction while maintaining a bid graph equivalent to the original. Further, we demonstrate how such modifications can be used to find valid item graphs of smaller treewidth.

4.1 Relating Item Graphs to Bid Graphs

Are item graphs and bid graphs entirely different, with no concrete means of mapping from one representation to the other, or are they related enough to allow us to find reductions between the two? Item graphs offer more flexibility than bid graphs, as shown in Chapter 2, Section 2.3.4, when we demonstrated multiple item graphs that could represent the same combinatorial auction, in addition to multiple combinatorial auctions represented by a single item graph. Expressing a combinatorial auction as an item graph appears to be different from expressing that same combinatorial auction as a bid graph, and we expect that any algorithm for solving WDP on an item graph is not going to easily translate to an algorithm on a bid graph. However, in this chapter we are mainly interested in how the item graph and bid graph are structurally related.

There are a few algorithms published in the literature for solving WDP that deal directly with an item graph representation of a combinatorial auction. Sandholm and Suri show that given a valid item graph, if the graph is a tree or a cycle then we can solve WDP in polynomial time [48]. In more detail, if a cycle is a valid item graph of a combinatorial auction, the winners can be determined optimally by solving several instances of an item graph that is a path, and removing bids that are no longer allowed [48]. The work is in attempting to find the correct edge to remove from the cycle, in order to construct a line

graph that gives an optimal solution to WDP for the cycle item graph. Recall that $dab(B)$, which we defined in Section 2.3.3, is the number of distinct atomic bids in the combinatorial auction. We consider when a cycle is a valid item graph of a combinatorial auction, and label the cycle as $C = 1, 2, 3, \ldots, |I|, 1$. We let item $|I| + 1$ be identified with item 1. For any integer ℓ, where $\ell \in \{1, 2, \ldots, |I|\}$, we say that an atomic bid $B = (S, p)$ *crosses the interval* $[\ell, \ell + 1]$ if $\{\ell, \ell + 1\} \subseteq S$ and $S \subset I$. If we let s denote the smallest number of distinct atomic bids that cross any interval of the form $[\ell, \ell + 1]$, where $\ell \in \{1, 2, \ldots, |I|\}$, then WDP on a combinatorial auction with a cycle as a valid item graph can be computed in worst-case time $O(\min\{s, |I|\}(dab(B) + |I|))$ [48].

When the valid item graph of a combinatorial auction is a tree, Sandholm and Suri solve WDP in worst-case time $O(dab(B)|I|)$ [48]. They choose a root node of the tree arbitrarily, and assign each vertex of the tree a *level*, which is its distance from the root. The root node is considered to have level 0. The *level* of a bid is the smallest level of any item in the bid. The algorithm by Sandholm and Suri uses dynamic programming to compute the optimal solution at each vertex of the tree in decreasing order of level [48]. As we will see in this section, the bid graph of a combinatorial auction with a tree as a valid item graph is chordal. The maximum weighted independent set of a chordal graph can be found in time that is linear in the number of vertices and edges in the bid graph [13].

For a given combinatorial auction, if a line or a cycle is a valid item graph for the auction, then we can construct the item graph in polynomial time [30, 12]. Conitzer *et al.* provide an algorithm for constructing a valid item graph of a combinatorial auction that is a tree, should one exist [7]. In addition, Conitzer *et al.* give a result on item

graphs of bounded treewidth, which happens to be a parameterized instance of WDP on combinatorial auctions [7].

Theorem 4.1 (Conitzer, Derryberry, and Sandholm, [7])**.** *Suppose we are given a combinatorial auction problem instance, together with a tree decomposition T with treewidth tw of an item graph G_I. Then the optimal solution to WDP can be determined in $O(|T|^2(dab(B)+1)^{tw+1})$ using dynamic programming.*

Consider the following parameterized version of WDP on combinatorial auctions:

tw, k-ITEM WINNER DETERMINATION (IWD(tw, k))

Input:	A set of agents A, items I, bids $B = B_1, B_2, \ldots, B_{\|A\|}$, item graph G_I which is valid for A, I and B, and a tree decomposition T of G_I with treewidth at most tw.
Parameters:	Positive integers k and tw.
Question:	Does there exist a set of mutually disjoint atomic bids $\{(S_1, p_1), (S_2, p_2), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell}(p_j) \geq k$?

Theorem 4.1 implies that we have an algorithm for IWD(tw, k) with running time $O(|T|^2(dab(B)+1)^{tw+1})$. Unfortunately, because the base of the term where tw is the exponent involves another part of the input, namely $dab(B)$, we cannot conclude that IWD(tw, k) is fixed-parameter tractable. In fact, we cannot claim it is either fixed-parameter tractable or intractable.

The solution presented by Conitzer *et al.* is vastly different from any algorithm designed to work on bid graphs, and it is beyond the scope of this thesis whether or not there is some relationship between the two, in the most general sense. If we consider only combinatorial

auctions that have item graphs of treewidth one, then the item graph is a forest, for which we can solve WDP on each component separately. As stated, we know of a polynomial-time solution to WDP for an item graph that is a tree [48], which easily extends to an item graph that is a forest. We are interested in investigating how, for certain fixed values of tw, the item graph relates to the bid graph of any possible combinatorial auction that has such an item graph representation. That is, given an item graph with $tw = 1$, say, we wish to conclude something about the bid graph expression of all combinatorial auctions that have a valid item graph representation with a treewidth of one.

Given an item graph of treewidth one, consider any combinatorial auction for which the item graph is valid. For this auction, we cannot place restrictions on the number of items in a bid, the number of bids, or the number of edges incident to a bid in the auction's bid graph. Namely, adding a bid whose bundle of items contains all items in I does not change the validity of the item graph if it is a tree. Further, the number of possible bids is equivalent to the number of subtrees of the item graph, which is exponential in the number of items. Finally, in the auction's bid graph, the number of bids to which a bid containing all items is incident is equal to one less than the number of bids. Thus, the number of edges incident to a bid in the bid graph is possibly exponential in the number of items. Therefore, a fixed-parameter reduction to any of the problems outlined in Chapter 3 is most likely complicated, as there does not appear to be any correlation between their parameters and the parameters in IWD(tw, k). As a result, we turn our attention away from fixed-parameter reductions and toward more general relationships between item graphs and bid graphs. Rather than starting by investigating item graphs of treewidth one, we will consider a simpler example where our item graphs are paths. First, however, we discuss whether or

not the item graph is connected.

Lemma 4.2. *If it is valid for an item graph G_I of a given combinatorial auction to be disconnected, then either the bid graph of the auction must be disconnected, or the combinatorial auction contains only bids on items in one component of G_I.*

Proof. Consider any bid B_1, which must contain only items from one connected component of G_I. Any other bid B_2 that is adjacent to B_1 in the bid graph must contain at least one item from B_1. Therefore, B_2 must contain only items from the same connected component of the item graph as B_1. If it did not, then the item graph would not be valid as the induced subgraph of G_I on the items of B_2 would be disconnected. If the bid graph is connected, then there is a path from B_1 to any other bid, and hence all bids must contain only items from the same connected component of G_I as B_1. If all bids contain only items from the same component of G_I, then this is valid. If not, we have a contradiction. □

We will assume that our item graph representation is connected, as otherwise our proofs will work on each of the connected components of the item graph.

Observation 4.3. *If a path is a valid item graph for a given combinatorial auction, then the bid graph representation of the auction is an interval graph.*

Proof. Let us consider an arbitrary combinatorial auction that has a valid item graph representation as a path. Each bid of the combinatorial auction must induce a connected subgraph of the path. The path has length $|I|$. Consider one of the vertices of degree one, u. Starting at u and walking along the path, without repeating any edge or vertex, relabel each vertex starting with the integer 1, and increasing the label value by 1 with

each subsequent vertex. To be more specific, the vertex adjacent to u will have label 2; the next vertex in the path will have label 3, and so on. With this labeling, it is easy to see that each bid in the combinatorial auction is a sub-interval of the interval $[1, |I|]$. The bid graph is then an intersection graph of the sub-intervals of the interval $[1, |I|]$, and therefore is an interval graph. □

Observation 4.3 was also noted by Sandholm and Suri, although no proof was given [48]. As a corollary to Observation 4.3, we see that if a cycle is a valid item graph, then the bid graph is a circular arc graph. A cycle can be viewed as a circle, with each bid an arc of the circle. The bid graph is then the intersection graph of those arcs, and hence is a circular arc graph. This observation was also made by Sandholm and Suri [48]. The maximum weighted independent set of a circular arc graph can be found in time $O(l|\mathcal{S}^*|)$, where l is the minimum number of arcs passing through some point on the circle [49]. This is comparable to the algorithm by Sandholm and Suri mentioned above, which has a running time of $O(\min\{s, |I|\}(dab(B) + |I|))$ and applies to valid item graphs that are cycles.

Drawing conclusions from an item graph with treewidth one is more complicated. First of all, a graph has treewidth one if and only if it is a forest. Since we are assuming that our item graph is connected, then our item graph must be a tree.

Theorem 4.4. *For a given combinatorial auction CA, if a graph $G_I = (V_I, E_I)$ of treewidth one is a valid item graph for the auction, then its bid graph representation is a chordal graph.*

This theorem follows directly from a theorem by Gavril [16]. The intersection graph of

a family of subtrees of a tree is called a *subtree graph*, where two subtrees intersect if they share a vertex [16]. Theorem 4.4 follows directly from a result by Gavril, which states that a graph is chordal if and only if it is a subtree graph [16]. Since G_I is a tree and by definition the bids of the combinatorial auction CA it represents induce connected subgraphs of G_I, the bids induce subtrees of G_I. By definition, the bid graph is the intersection graph of the bids of CA, and thus is also the intersection graph of a family of subtrees of G_I. Therefore, Gavril's result applies and the bid graph must be chordal [16].

We note that we cannot use the other direction of Gavril's result to make Theorem 4.4 an "if and only if" statement. Gavril places no restrictions on the structure of the subtree graph for which the result applies, while we require that the vertices are the set of items in order to be a valid item graph for a given bid graph. In Section 4.2, we will introduce a notion of bid graph equivalence in order to make full use of Gavril's result. For now, we continue to investigate some relationships between bid graphs and item graphs without allowing modifications to the auction.

Theorem 4.4 demonstrates that we have a relationship in one direction between item graphs that are trees and bid graphs. Namely, for a given combinatorial auction, if a tree is a valid item graph for the auction, then its bid graph representation is a chordal graph. It is not immediately obvious whether or not all combinatorial auctions whose bid graphs are chordal must have valid item graphs that are trees. To investigate this question, we aim to prove that a combinatorial auction's bid graph must be chordal if it has a valid item graph with maximum cycle length three. To clarify, we say a graph has *maximum cycle length* k if all cycles in the graph have length at most k.

The proofs of Lemma 4.5 and Theorem 4.7 focus on one arbitrary cycle of length three

in our item graph $G_I = (V_I, E_I)$. Without loss of generality, we label the vertices involved as 1, 2, and 3. Consider the subgraph of G_I constructed by removing edges $\{1,2\}$, $\{2,3\}$ and $\{1,3\}$. This subgraph must have exactly three connected components because the maximum cycle length in G_I is three. Therefore, there exist connected subgraphs $G_i = (V_i, E_i)$, $1 \leq i \leq 3$, of G_I such that $V_I = \bigcup_{i=1}^{3} V_i$, $E_I = (\bigcup_{i=1}^{3} E_i) \cup \{\{1,2\},\{2,3\},\{1,3\}\}$, $i \in V_i$ for $1 \leq i \leq 3$, $E_i = \{(u,v) \in E_I \mid u,v \in V_i\}$ for $1 \leq i \leq 3$, and for all $1 \leq i < j \leq 3$, $V_i \cap V_j = \emptyset$. See Figure 4.1 for a depiction of this item graph. We refer back to Figure 4.1 in the proofs of Lemma 4.5 and Theorem 4.7.

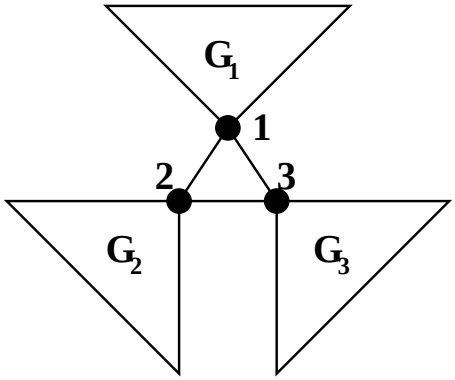


Figure 4.1: Example of an item graph with at least one cycle of length three. Subgraph G_i is assumed to have maximum cycle length three.

In Theorem 4.7, we prove that a combinatorial auction's bid graph must be chordal if it has a valid item graph with maximum cycle length three. We first prove that if a tree with one extra edge resulting in a cycle of length three is a valid item graph for a combinatorial auction, then the auction's bid graph must be chordal. In both proofs, we use a lemma about combinatorial auctions when their bid graphs are chordal. Lemma 4.5 was derived from the idea of removing an edge in any cycle of length three from a valid item graph,

and then adding that edge back. Of course, with the removal of an edge in the item graph, we can no longer have certain bids and so these bids are removed to result in a modified combinatorial auction. We show that given any cycle, if for each edge in that cycle the modified combinatorial auction has a chordal bid graph, then so does the original auction.

For brevity, we say a bid (S, p) *contains item set* $S' \subseteq I$ if $S' \subseteq S$. Further, we say a bid (S, p) *does not contain item set* $S' \subseteq I$ if $S' \not\subset S$. We introduce notation to denote when a bid contains a set of items S^+, while not containing another set of items S^-. For all $1 \leq i \leq |I|$, we let i^- denote an alternate label for item i. Hence, label i^- represents the same item as i. The label i is used to denote items that are contained in a bid, whereas the label i^- is used to denote items that are not contained in a bid. We let $I^- = \{1^-, 2^-, 3^-, \ldots, |I|^-\}$, and let $S^* = S^+ \cup S^-$, where $S^+ \subseteq I$, $S^- \subseteq I^-$, and $\{i \mid i \in S^+, i^- \in S^-\} = \emptyset$. We say a bid (S, p) *contains item set* $S^* = S^+ \cup S^-$ if $S^+ \subseteq S$ and $\{i \mid i \in S, i^- \in S^-\} = \emptyset$. For example, we define bid $B = (S, p)$ by $S = \{1, 2, 4\}$ and p arbitrary. Bid B contains item sets $\{1, 2\}$, $\{1, 4\}$, $\{1, 2, 4\}$, and $\{1, 3^-, 4\}$. Bid B does not contain item set $\{4^-\}$, $\{1, 2, 3\}$, or $\{1, 2^-, 3^-, 4\}$.

Lemma 4.5. *We let CA denote a combinatorial auction that has a valid item graph $G_I = (V_I, E_I)$ such that the maximum length of any cycle in G_I is three and G_I has at least one cycle. We denote the bid graph of CA by G_B. We let C_I be any cycle in G_I, and we label its vertices as 1, 2, and 3. We let $\{i, j\}$ be any edge in cycle C_I, and CA' be the modified combinatorial auction obtained by removing all bids containing $\{i, j\}$, but not containing $\{1, 2, 3\}$, from CA. Given any cycle C_I in G_I, if for any edge $\{i, j\}$ and CA' as defined above the bid graph of CA' is chordal, then the bid graph of CA is chordal.*

Proof. We refer the reader to Figure 4.1 for a depiction of G_I, as previously discussed.

We assume that for any $\{i, j\}$ in C_I, the combinatorial auction resulting from the removal of all bids containing $\{i, j\}$, but not containing $\{1, 2, 3\}$, has a chordal bid graph. For each $\{i, j\}$, the resulting combinatorial auction may be different, but they all have a chordal bid graph. Without loss of generality, let CA' be the combinatorial auction resulting from the removal of all bids containing $\{1, 2, 3^-\}$ from CA. The bid graph of CA' is chordal. We have only to consider what happens when we add back all the removed bids containing $\{1, 2, 3^-\}$, and prove that the result is a chordal bid graph.

Assume that the bid graph is no longer chordal after adding back either one or two bids containing $\{1, 2, 3^-\}$. We do not consider adding back more bids because any chordless cycle can involve no more than two bids containing $\{1, 2, 3^-\}$. Specifically, by our definition of a bid graph in Section 2.3.3, if two bids share an item then they are adjacent in G_B. Therefore, if a cycle involves three bids containing $\{1, 2, 3^-\}$ then by definition they must all be pairwise adjacent to one another. Hence, the cycle cannot be chordless. If we are assuming that the bid graph is no longer chordal, then it must have at least one cycle C_B of length at least four with no chords. Cycle C_B must involve at least one bid containing $\{1, 2, 3^-\}$ because the original bid graph was chordal, but not more than two such bids because otherwise it would not be chordless. We note that the bids that we add back do not contain item 3.

We claim that cycle C_B must also involve a bid containing item set $\{1^-, 2, 3\}$. If not, we construct a bid graph G'_B by removing all bids containing $\{1^-, 2, 3\}$ from G_B. None of the bids removed are from C_B because C_B does not involve a bid containing item set $\{1^-, 2, 3\}$. The resulting bid graph is not chordal because the original bid graph was not chordal and we removed none of the bids in the chordless cycle C_B. However, by assumption, the bid

graph must be chordal, which is a contradiction. By the same logic, the cycle C_B must involve a bid containing $\{1, 2^-, 3\}$. Specifically, we can derive another contradiction; the bid graph resulting from removing all bids containing $\{1, 2^-, 3\}$ from G_B is not chordal by construction, but was assumed to be chordal.

Therefore, the cycle C_B must have the following three distinct bids: one containing $\{1, 2, 3^-\}$, one containing $\{1^-, 2, 3\}$, and another containing $\{1, 2^-, 3\}$. These three bids form a clique of size three within C_B, which is not possible because C_B is chordless. Therefore, the bid graph resulting in the addition of one or two bids containing both items 1 and 2 (but not containing item 3) must be chordal.

We continue to add back bids containing $\{1, 2, 3^-\}$ two at a time until only one remains. Finally, we add the last remaining bid containing $\{1, 2, 3^-\}$. From above, at no point during the addition of one or two bids containing $\{1, 2, 3^-\}$ can the resulting bid graph become non-chordal. Therefore, the bid graph resulting from adding back all bids containing $\{1, 2, 3^-\}$ is chordal. Hence, the original bid graph G_B is chordal. □

With the result of Lemma 4.5, we prove Lemma 4.6, which will be used as a base case of Theorem 4.7.

Lemma 4.6. *For a given combinatorial auction CA, if a tree with one extra edge resulting in a cycle of length three is a valid item graph for the auction, then its bid graph representation is a chordal graph.*

Proof. We denote the valid item graph as G_I and the bid graph as G_B. Because G_I is a tree with one extra edge resulting in a cycle of length three, we can depict the graph as in Figure 4.1. For simplicity and without loss of generality, we have labeled the items of the

cycle as 1, 2 and 3. Subgraphs G_1, G_2 and G_3 of G_I are trees. Consider the item graph resulting from the removal of edge $\{1,2\}$, $\{2,3\}$, or $\{1,3\}$. This item graph is valid for a modified combinatorial auction resulting from the removal of all bids containing $\{1,2,3^-\}$, $\{1^-,2,3\}$, or $\{1,2^-,3\}$, respectively, from CA. From Theorem 4.4, the bid graph of such an auction is chordal.

We now apply the result of Lemma 4.5. Given any cycle (in this case there is only one) and for each edge $\{i,j\}$ in that cycle, the modified combinatorial auction resulting from the removal of all bids containing $\{i,j\}$, but not containing $\{1,2,3\}$, from CA has a chordal bid graph. Therefore, by Lemma 4.5 the original combinatorial auction CA has a chordal bid graph. □

A natural extension of the result of Lemma 4.6 is the following theorem.

Theorem 4.7. *For a given combinatorial auction, if there exists a valid item graph $G_I = (V_I, E_I)$ representing the auction such that the maximum length of any cycle in G_I is three, then the bid graph for the auction is chordal.*

Proof. For this proof, when we refer to a valid item graph, the length of all cycles in the graph is three. Theorem 4.4 handles the case when there are no cycles, so we assume that there is at least one cycle of length three.

The proof is by induction on the number of cycles of length three. Lemma 4.6 proves the base case, when the number of cycles of length three is one. We assume the result holds for all valid item graphs with at most k cycles of length three. We will show the result holds for all valid item graphs with $k+1$ cycles of length three.

Consider a valid item graph that has $k+1$ cycles of length three. We focus on one

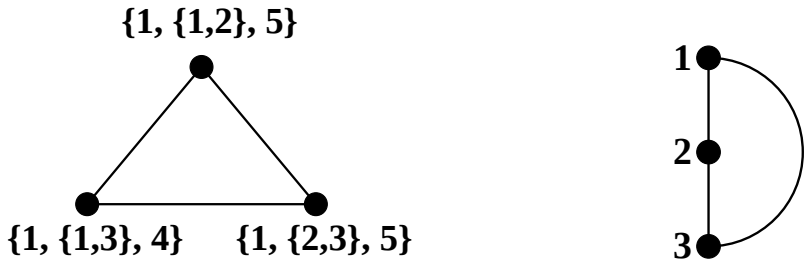
arbitrary cycle C_I of length three in G_I, and without loss of generality, we label the vertices involved as 1, 2, and 3. Recall Figure 4.1 for a depiction of G_I, as previously discussed.

If we remove any edge $\{i, j\}$ in C_I from the item graph, and remove all bids from the combinatorial auction that contain $\{i, j\}$, but not $\{1, 2, 3\}$, then by construction the resulting item graph is valid for the new auction. By our inductive hypothesis, the new combinatorial auction's bid graph is chordal because the number of cycles in the item graph decreased by one. Therefore, by Lemma 4.5 the original combinatorial auction's bid graph is chordal. □

As a result of Lemma 4.6 and Theorem 4.7, it appears that we can conclude that if a bid graph of a combinatorial auction is chordal, then a tree is not necessarily a valid item graph for the auction. This would imply that our result in Theorem 4.4 cannot be modified into an "if and only if" statement. To show this negative result, we prove that there exists a combinatorial auction with a chordal bid graph such that the auction does not have a valid item graph that is a tree.

The cycle of length three in Figure 4.2(a) is a chordal bid graph, and the item graph in Figure 4.2(b) is the only valid item graph for such a bid graph. Thus we conclude that if a bid graph of a combinatorial auction is chordal, then a tree is not necessarily a valid item graph for the auction.

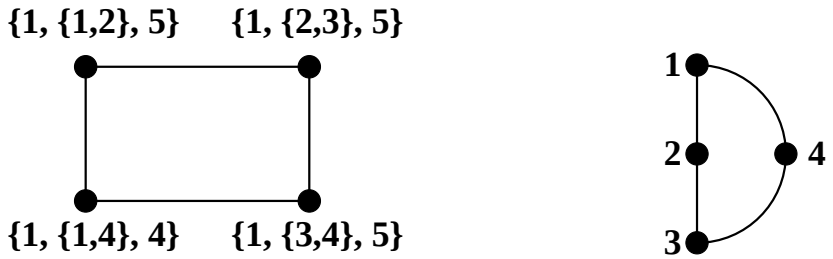
To conclude our discussion on the relationships between item graphs and bid graphs, we consider the combinatorial auction with the following atomic bids, which has a valid item graph of treewidth two: $(\{1, 2\}, 5)$, $(\{2, 3\}, 5)$, $(\{3, 4\}, 5)$, and $(\{1, 4\}, 4)$. The bid graph is a cycle of length four, and a cycle of length four is also a valid item graph, as depicted in Figure 4.3. This example is interesting because the item graph has treewidth



(a) (b)

Figure 4.2: Example of (a) a bid graph that is a cycle of length three, and (b) its only valid item graph.

two, which means that given a combinatorial auction for which an item graph of treewidth two is valid, the auction's bid graph is not necessarily a chordal graph. Therefore, the result of Theorem 4.4 does not extend to valid item graphs of treewidth two.



(a) (b)

Figure 4.3: Example of (a) a non-chordal bid graph, and (b) one of its valid item graphs.

We leave open the general relationship between item graphs of treewidth two and bid graphs. Note that whatever classification of bid graphs may arise from such a relationship, the classification must encompass the class of chordal graphs, because any item graph of treewidth one is also considered to have treewidth two, by the definitions of treewidth and tree decompositions.

We have concluded our discussion of the relationship between item graphs and bid graphs for fixed combinatorial auctions. In Section 4.2 we begin discussing the modification of combinatorial auctions. Specifically, we investigate modifying combinatorial auctions while maintaining essentially the same bid graph, in order to achieve an item graph of lower treewidth.

4.2 Modifying the Combinatorial Auction for Smaller Treewidth

We modify combinatorial auctions while maintaining equivalent bid graphs, which we define momentarily, and explain why such modifications are useful. The motivation behind studying the modification of combinatorial auctions begins with the fact that item graphs are hard to construct. As shown by Conitzer *et al.*, constructing a valid item graph with the fewest edges is NP-complete [7]. Further, this hardness result holds even when an item graph of treewidth two exists, each bid is on at most 5 items, and we do not necessarily require the item graph constructed to be of treewidth two [7]. Constructing an item graph of fixed treewidth may also be hard. Given a combinatorial auction, Conitzer *et al.* construct a valid item graph of treewidth one in polynomial time, if the graph exists [7]. However, as shown by Gottlob and Greco, it is NP-hard to decide whether or not a combinatorial auction has a valid item graph of treewidth three [18].

Throughout the literature, it is assumed that a combinatorial auction is fixed before the construction of a valid item graph. Modifying the combinatorial auction in order to achieve a valid item graph of smaller treewidth is not discussed in this context. We

investigate this issue in this section and with a specific definition of equivalence among combinatorial auctions and apply a theorem by Gavril, which suggests a new method for constructing valid item graphs of bounded treewidth that may lead to a fixed-parameter tractable algorithm for constructing item graphs [16].

To begin, we define bid graph equivalence. Intuitively, two bids graphs are equivalent if they are isomorphic, and two combinatorial auctions are equivalent if their bid graphs are also equivalent. We now give the formal definition.

Definition 4.8 (Bid Graph Equivalence)**.** *Given two bid graphs, $G = (V, E)$ and $G' = (V', E')$, we say that G is* equivalent *to G' (and vice-versa) if there exists an isomorphism f of G and G' and for all $v = (S, p) \in V$, $f(v) = (S', p') \in V'$ we have $p' = p$. If two combinatorial auctions have equivalent bid graphs then we say the auctions are* equivalent*.*

Our modification of G will only involve the bundles of items represented by the vertices, and those bundles of items will be modified in a way that does not add any edges to G, or remove any edges from G during the process. Thus, our isomorphism f mapping V to V' is the identity mapping. Therefore for any $v \in V$, while the bundle of items associated with $f(v)$ might differ from the bundle associated with v, the weights of the vertices are the same.

Restricting our modifications to the above is important so that we can guarantee that the optimal solution to WDP is the same for both graphs. That is, the maximum weighted independent set on both graphs is identical because we are working with equivalent graphs where the weight of vertex $v \in V$ is the same as that of vertex $f(v) \in V'$. Adding and removing items are only ways to modify a combinatorial auction so that the new bid graph remains isomorphic to the old bid graph. We cannot add or remove bids, as this would

change the number of vertices in the bid graph. As we noted earlier, adding bids of value 0 can help in some cases, but our result does not require these special bids.

Theorem 4.9. *We let CA denote a combinatorial auction that has a valid item graph $G_I = (V_I, E_I)$. If the bid graph G_B of CA is chordal, then there exists a combinatorial auction CA' equivalent to CA that has a tree G'_I as a valid item graph.*

This theorem follows directly from a theorem by Gavril [16]. We recall from Section 4.1 that the intersection graph of a family of subtrees of a tree is called a *subtree graph*, where two subtrees intersect if they share a vertex [16]. Theorem 4.9 follows directly from a result by Gavril, which states that a graph is chordal if and only if it is a subtree graph [16]. Since G'_I is a tree and by definition the bids of the combinatorial auction CA' it represents induce connected subgraphs of G'_I, the bids induce subtrees of G'_I. By definition, the bid graph is the intersection graph of the bids of CA', and thus is also the intersection graph of a family of subtrees of G'_I. Therefore, Gavril's result applies and the bid graph of CA' must be chordal. Since CA' is equivalent to CA, their bid graphs are isomorphic and thus the bid graph of CA must also be chordal.

It is important we note that with our definition of combinatorial auction equivalence, Gavril has shown that even if the smallest treewidth for G_I is arbitrarily larger than one, there still exists a combinatorial auction equivalent to CA with a tree as a valid item graph, given that the bid graph of CA is chordal. This perspective on combinatorial auction equivalence and the existence of item graphs of small treewidth has never been shown before.

Corollary 4.10. *For a given combinatorial auction CA, the bid graph of the auction is*

chordal if and only if there exists a combinatorial auction that is equivalent to CA for which a graph of treewidth one is a valid item graph.

Gavril gives an efficient algorithm for constructing an item graph $G_I = (V_I, E_I)$ that is a tree for any given bid graph $G_B = (V_B, E_B)$ that is chordal [16]. We let CA be the combinatorial auction represented by G_B. Gavril shows that $|V_I| \leq |V_B| = dab(B)$ and that G_I can be found in at most $O(dab(B)^4)$ time [16]. The construction considers each maximal clique C in G_B, of which there are at most $dab(B)$ because G_B is chordal, and constructs a vertex in G_I that represents C [16]. We will see that this item graph is valid for a combinatorial auction that is equivalent to CA.

We consider the vertices of G_I as items. We copy the bid graph G_B (to ensure our bid graphs are equivalent), and change the contents of every bid B_i of our copy to be exactly the set of items of G_I that represent all the cliques to which B_i belongs. Specifically, B_i is now a bid on items $C_1, C_2, \ldots, C_k$, where C_j is a vertex of G_I representing a clique of G_B, which contains vertex B_i. The resulting bid graph is equivalent to G_B and by construction is a subtree graph of G_I. Therefore, G_I is a valid item graph for a combinatorial auction that is equivalent to CA. Therefore, if we find any solution to WDP using the newly constructed item graph G_I, this solution can be easily translated to a solution for the original combinatorial auction.

With the result of Theorem 4.9, it becomes difficult to gauge the usefulness of item graphs. If it is possible for a combinatorial auction to have a valid item graph of treewidth one while another equivalent combinatorial auction does not, then it is possible that we are translating our auction to a less efficient form. Further, there may or may not exist a combinatorial auction with a chordal bid graph for which all valid item graphs have

treewidth at least tw, for some large tw. That is, it is unclear how big of an improvement is possible using the equivalent combinatorial auction technique. We have seen an example of a combinatorial auction with a chordal bid graph for which all valid item graphs have treewidth at least two, but the improvement could be much larger. Given this potentially large lack of consistency between item graph representations of equivalent combinatorial auctions, the construction process, as previously described in the literature, appears to be flawed.

Currently the item graph construction process, for which the results by Conitzer *et al.* and Gottlob and Greco hold, involves fixing the combinatorial auction and finding a suitable item graph [7, 18]. We have seen that this is not necessarily optimal, although we leave open whether or not Gavril's construction process can be extended beyond using chordal bid graphs to find an item graph of smallest treewidth in polynomial time in cases where the bid graph is not chordal. Since it is NP-hard to find a valid item graph of treewidth tw, for $tw \geq 3$ [18], it may be better to find equivalent combinatorial auctions for which finding a valid item graph of treewidth tw is easy. We leave open whether such an algorithm is fixed-parameter tractable.

As shown by Gottlob and Greco, it is NP-hard to decide if a combinatorial auction has a valid item graph of treewidth three [18]. This suggests that Theorem 4.1 on item graphs of bounded treewidth tw has no practical use, because such graphs are hard to construct for any fixed $tw \geq 3$. However, Theorem 4.9 leaves open the possibility of altering the construction process of valid item graphs so that the item graph representation becomes useful. With the result by Gavril [16], we have shown only one possible alteration (using the cliques of the bid graph), but there may be many others that can be applied to construct

valid item graphs of bounded treewidth. For a given combinatorial auction, it may be fixed-parameter tractable to decide if there exists another, equivalent combinatorial auction that has a valid item graph of treewidth tw. If this is the case, and constructing the valid item graph of treewidth tw is also fixed-parameter tractable, then these item graphs can be used in practice to solve the winner determination problem for combinatorial auctions. If the decision problem or construction process is NP-hard, then item graphs of bounded treewidth have no practical use.

Chapter 5

Conclusions and Future Work

During the course of this thesis, we examined combinatorial auctions (CAs) in detail. Combinatorial auctions (CAs) are important tools for allocating items to agents who wish to communicate preferences over bundles of items. The process of determining the allocation of items is known as the winner determination problem (WDP) [35, 1, 32, 9, 8]. Finding a solution to the winner determination problem (WDP) is NP-complete in the general case, and it is also equally difficult to find approximate solutions. We examined previous algorithms for optimally solving WDP for general CAs, and also reviewed some solutions to WDP for CAs when the auction was restricted. Under special circumstances that enforce structure on the CA, WDP can become easier to solve. Finally, we looked at the relationship between two different representations of combinatorial auctions; namely, item graphs and bid graphs. During our investigation of the relationship between item graphs and bid graphs, we found that if a given combinatorial auction has a valid item graph of treewidth one, then the bid graph of the auction must be chordal. Further, by considering a new

technique for constructing valid item graphs, we were able to characterize combinatorial auctions with chordal bid graphs by showing that these combinatorial auctions must each have an equivalent combinatorial auction with a valid item graphs of treewidth one.

To further discuss our results, we recall the definition of WD(k) from Section 3.1. In Section 3.1, WD(k) was shown to be $W[1]$-hard via a reduction from a $W[1]$-hard parameterization of independent set.

k-WINNER DETERMINATION (WD(k))

Input: A set of agents A, items I, and bids $B = B_1, B_2, \ldots, B_{|A|}$.

Parameter: Positive integer k.

Question: Does there exist a collection of mutually disjoint atomic bids $\{(S_1, p_1), (S_2, p_2), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell} p_j \geq k$?

One of the main contributions of this thesis is its demonstration of the use of parameterized complexity in the investigation of WDP for CAs. With parameterized complexity theory, it is possible to parameterize WDP and gain insight into new algorithms, and prove when parameterizations are as hard as any solution to WD(k). We hope that this methodology will open up questions and ideas that lead to improved ways of solving and structuring WDP for CAs.

In Chapter 3 we made use of a new approach in analyzing the complexity of structured versions of WDP. By using parameterized complexity theory, we were able to formulate parameterizations of WDP and formally show when these parameterized problems can be solved by fixed-parameter intractable algorithms. Further, in four of our parameterizations of WDP, we were able to demonstrate fixed-parameter tractability.

We recall that the problem WD(β, k), as presented in Section 3.3, is defined similarly

to WD(k), except that agent i's bid graph belongs to graph class β. For the main result of Chapter 3, we found a definition of graph class β such that the resulting WD(β, k) problem is $W[1]$-hard. We assumed that the definition of graph class β is not a function of any part of the input to WD(β, k) that is unknown before the start of the combinatorial auction. Then, if β has a minimal β graph of size at most a polynomial in the size of the input to WD(β, k), which imposes no restriction on the interaction between the atomic bids of different agents, then the resulting WD(β, k) problem is $W[1]$-hard. This restriction of β is lax enough to allow many different graph classes. It is useful because in the investigation of CAs as they apply to specific economic areas, one may find structure in the bid graphs of individual agents. With our result, we now have additional examples of when a structured bid graph will not help us solve WDP more efficiently. Knowing structures that are *not* helpful is often just as important as finding ones that lead to fixed-parameter tractability.

In Chapter 4, we investigated the relationship between item graphs and bid graphs. We first introduced a one-directional relationship between item graphs and bid graphs. Given a combinatorial auction, we presented a result by Gavril which shows that if a tree is a valid item graph for the auction, then the auction's bid graph must be chordal [16]. We then turned our discussion to combinatorial auctions whose bid graphs are chordal, and found a subset of valid item graphs for equivalent combinatorial auctions that share a common property. Our discussion involved a new technique for constructing valid item graphs, where by adding items to the combinatorial auction we can maintain an equivalent combinatorial auction while simplifying a valid item graph. With this method, we were able to use a result by Gavril, which states that if a given combinatorial auction has a chordal bid graph, then there exists an equivalent combinatorial auction with a tree as a

valid item graph [16].

One of the most important contributions of this thesis comes from Chapter 4. The new technique for constructing item graphs using our new definition of combinatorial auction equivalence has not been previously presented. The result by Gottlob and Greco [18], which shows that it is NP-hard to decide whether or not a combinatorial auction has a valid item graph of treewidth three, brings to question the practicality of studying item graphs of bounded treewidth. The new construction technique that we introduce using a theorem by Gavril opens another avenue of investigation for determining how useful item graphs of bounded treewidth are in practice [16].

Conitzer *et al.* introduced an algorithm for solving WDP for combinatorial auctions whose item graphs have treewidth at most tw [7]. This algorithm implicitly defines a parameterized version of the winner determination problem, which we presented in Chapter 4 and now recall.

tw, k-ITEM WINNER DETERMINATION (IWD(tw, k))

Input:	A set of agents A, items I, bids $B = B_1, B_2, \ldots, B_{\|A\|}$, item graph G_I which is valid for A, I and B, and a tree decomposition T of G_I with treewidth at most tw.
Parameters:	Positive integers k and tw.
Question:	Does there exist a set of mutually disjoint atomic bids $\{(S_1, p_1), (S_2, p_2), \ldots, (S_\ell, p_\ell)\}$ such that $\sum_{j=1}^{\ell}(p_j) \geq k$?

The algorithm presented by Conitzer *et al.* [7], while polynomial for fixed tw, does not demonstrate that IWD(tw, k) is fixed-parameter tractable. Further, we do not have a proof that IWD(tw, k) is fixed-parameter intractable. Ideally, if we could find a fixed-

parameter reduction from item graphs of fixed treewidth tw to a fixed-parameter tractable parameterization of WDP on combinatorial auctions, then we would be able to prove that IWD(tw, k) is fixed-parameter tractable. It would also suffice to provide a fixed-parameter reduction from a fixed-parameter intractable parameterization of WDP on combinatorial auctions to item graphs of fixed treewidth tw; this would show IWD(tw, k) to be fixed-parameter intractable. The problem of determining whether or not IWD(tw, k) is fixed-parameter tractable remains open.

Although we have introduced a new technique for constructing item graphs using a theorem by Gavril [16], we have not discussed the computational complexity of various aspects of this new technique. For a given combinatorial auction, it remains open whether or not it is fixed-parameter tractable to decide if there exists another, equivalent combinatorial auction that has a valid item graph of treewidth tw. For $tw = 1$, the decision problem is solved as we can simply verify that the combinatorial auction's bid graph is chordal, which can be done in time that is a polynomial in the size of the auction [50]. Also, for a given combinatorial auction CA, if there exists an equivalent combinatorial auction that has a tree as a valid item graph, then a tree can be constructed in at most $O(dab(B)^4)$ time that is a valid item graph for some combinatorial auction that is equivalent to CA [16]. However, the computational hardness of the construction problem is open for general treewidth $tw > 1$. Specifically, for a given combinatorial auction, if there exists an equivalent combinatorial auction CA that has a valid item graph of treewidth at most tw, then we do not know whether or not it is NP-hard to construct an item graph of treewidth at most tw that is valid for CA.

In addition to investigating the computational complexity of the construction technique

introduced in Chapter 4, one may wish to investigate relationships between item graphs of treewidth tw and bid graphs, for $tw > 1$. Our new definition of combinatorial auction equivalence and the resulting construction technique derived from Gavril's result may be useful in this investigation. Further, Gottlob and Greco recently introduced a new method for qualifying hypergraphs of combinatorial auctions, which they refer to as hypertrees of bounded hypertree width [18]. One could investigate the relationships between combinatorial auctions with hypertrees of bounded hypertree width and the auctions' respective bid graphs. Item graphs of bounded treewidth are a special case of hypertrees of bounded hypertree width, and as such it would be interesting to see the parallels, if any, between their relationships with bid graphs. Despite the fact that item graphs of bounded treewidth are a special case of this new model, we can construct a hypertree of bounded hypertree width for a given combinatorial auction, should one exist, in time that is polynomial in the size of the auction [18]. A natural question would be to ask how does Gavril's construction technique affect the new model by Gottlob and Greco? Does the new construction method account for why an item graph of bounded treewidth was previously NP-hard to construct (even for treewidth three), while a hypertree of bounded hypertree width is polynomial to construct (for any fixed hypertree width)?

There are also a number of obvious avenues for future research involving parameterized complexity theory. These include finding additional parameterizations of WDP with respect to bid graphs, and with respect to item graphs. For instance, one might consider other structural restrictions for bid graphs, including additional graph classes, and then investigate WDP under these conditions. Solving WDP using bid graphs is equivalent to finding a maximum weighted independent set of a graph, which is a well-studied problem.

Hence, it is unlikely that additional contributions can be made by studying the maximum weighted independent set problem. It is more likely that parameterizing parts of the combinatorial auction itself, and then seeing how this affects the auction's bid graph, will lead to new results. With respect to item graphs, one may wish to investigate how parameterizing parts of a combinatorial auction affects item graph representations of the auction, both with fixed treewidth and without fixed treewidth.

Finally, one may wish to consider defining a notion of equivalence among item graphs. Using different definitions of equivalence, one may discover a relationship between bid graphs of combinatorial auctions with equivalent item graphs. Because there are potentially many item graphs for a single combinatorial auction, a definition of equivalence for item graphs may have to take this into consideration.

Bibliography

[1] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. *ICMAS*, pages 39–46, 2000.

[2] A. Archer, C. H. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *SODA*, pages 205–214, 2003.

[3] C. Boutilier, M. Goldszmidt, and B. Sabata. Sequential auctions for the allocation of resources with complementarities. In *IJCAI*, pages 527–534, Stockholm, Sweden, 1999.

[4] M. Cesati. Compendium of parameterized problems. http://bravo.ce.uniroma2.it/home/cesati/research/compendium/, September 2005.

[5] B. Chandra and M. M. Halldórsson. Greedy local improvement and weighted set packing approximation. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 169–176, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[6] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1), September 1971.

[7] V. Conitzer, J. Derryberry, and T. Sandholm. Combinatorial auctions with structured item graphs. In *AAAI*, pages 212–218, 2004.

[8] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[9] S. de Vries and R. Vohra. Combinatorial auctions: A survey. *Informs Journal on Computing*, 15(3):284–309, 2003.

[10] R. Diestel. *Graph Theory*. Springer-Verlag, Heidelberg, 2005.

[11] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in computer science. Springer, New York, 1999.

[12] E. M. Eschen and J. P. Spinrad. An $O(n^2)$ algorithm for circular-arc graph recognition. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 128–137, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[13] A. Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the Fifth British Combinatorial Conference*, pages 211–226, University of Aberdeen, Aberdeen, 1975. Congr. Numer., Vol. XV, Utilitas Math., Winnipeg, Man., 1976.

[14] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *IJCAI*, pages 548–553, 1999.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1983.

[16] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. of Comb. Theory, Series B*, 16:47–56, 1974.

[17] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16:539–548, 1964.

[18] G. Gootlob and G. Greco. On the complexity of combinatorial auctions: Structured item graphs and hypertree decompositions. In *ACM Conference on Electronic Commerce*, San Diego, CA, USA, 2007. ACM.

[19] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–31, July 1973.

[20] M. M. Halldórsson. Approximations of independent sets in graphs. In *APPROX '98: Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 1–13, London, UK, 1998. Springer-Verlag.

[21] M. M. Halldórsson. Approximation of weighted independent set and hereditary subset problems. *J. Graph Algorithms Appl.*, 4(1):1–16, 2000. Early versions appeared in: Computing and Combinatorics, Proceedings of the 5th Annual International Conference (COCOON), Tokyo, Japan 1999; and in: Lecture Notes in Computer Science, Vol. 1627, Springer, Berlin, 1999, pp. 261–270.

[22] M. M. Halldórsson, J. Kratochvíl, and J. A. Telle. Independent sets with domination constraints. *Discrete Appl. Math.*, 99(1-3):39–54, 2000. Also appeared in: Proceedings of the 25th International Conference on Automata, Languages, and Programming (ICALP), Aalborg, Denmark, Springer Lecture Notes in Computer Science, Vol. 1443, July 1998.

[23] M. M. Halldórsson and H. C. Lau. Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and coloring. *J. Graph Algorithms Appl.*, 1, 1997.

[24] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999.

[25] B. D. Hausch. Multi-object auctions: sequential vs. simultaneous sales. *Manage. Sci.*, 32(12):1599–1610, 1986.

[26] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Appl. Math.*, 6:243–254, 1983.

[27] D. S. Hochbaum. *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, Boston, MA, 1997.

[28] W. Jia, C. Zhang, and J. Chen. An efficient parameterized algorithm for m-set packing. *Journal of Algorithms*, 50:106–117, 2004.

[29] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, New York, NY, 1972. Plenum Press.

[30] N. Korte and R. H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1):68–81, 1989.

[31] I. Koutis. A faster parameterized algorithm for set packing. *Inf. Process. Lett.*, 94(1):7–9, 2005.

[32] V. Krishna. Auction theory. In *Auction Theory.* Academic Press, 2002.

[33] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[34] Daniel Lehmann, Liaden Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 96–102, New York, NY, USA, 1999. ACM Press.

[35] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford, 1995.

[36] R. P. McAfee and J. McMillan. Analyzing the airwaves auction. *Journal of Economic Perspectives*, 10(1):159–175, 1996.

[37] J. McMillan. Selling spectrum rights. *J. Economic Perspectives*, 8(3):145–162, 1994.

[38] P. Milgrom. Putting auction theory to work: The simultaneous ascending auction. *Journal of Political Economy*, 108(2):245–272, April 2000.

[39] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract. In *Eighteenth national conference on Artificial intelligence*, pages 379–384, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[40] N. Nisan. Bidding and allocation in combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 1–12, 2000.

[41] N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2006.

[42] N. Nisan and A. Ronen. Computationally feasible vcg mechanisms. *J. Artif. Intell. Res. (JAIR)*, 29:19–47, 2007. An earlier version appeared at the ACM Conference on Electronic Commerce, 2000, pp. 242–252.

[43] D. Parkes. Iterative combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2006.

[44] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Sci.*, 44(8):1131–1147, 1998.

[45] T. Sandholm. Issues in computational vickrey auctions. *Internal. J. Electronic Commerce*, 4(3):107–129, 2000. Special Issue on Applying Intelligent Agents for Electronic Commerce. A short, early version appeared at the Second International Conference on Multi-Agent Systems (ICMAS), 1996, pp. 299–306.

[46] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[47] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[48] T. Sandholm and S. Suri. BOB: Improved winner determination in combinatorial auctions and generalizations. *Artificial Intelligence*, 145:33–58, 2003. Earlier version: Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations. *AAAI-00*, pp. 90–97.

[49] J. P. Spinrad. *Efficient Graph Representations*, volume 19 of *Fields Institute Monographs*. American Mathematical Society, 2003.

[50] R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.

[51] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.