Algorithmic and Structural Complexities of Menus in Unit-Demand Auctions

Daniel Schoepflin Clayton Thomas S. Matthew Weinberg

December 7, 2025

Abstract

A player's menu—the set of all allocations she might receive given other players' reports—is a foundational concept in mechanism design. Menus capture the structure of strategyproofness: a mechanism is strategyproof if and only if each player always receives her favorite allocation from her menu. This leads to a number of natural questions: How can canonical mechanisms be described in terms of a player picking from her menu? How does one player picking from their menu affect other players' allocations and menus? In this paper, we investigate such questions for the canonical assignment problem with unit demand and transferable utility: the Unit Demand VCG auction (UD).

We study ascending and descending unit-demand auctions, a natural algorithmic model for calculating and describing auctions. We establish (i) a descending-price auction can calculate the outcome of UD while computing any particular bidder's menu along the way, while (ii) no ascending-price auction can do the same. We then study complexity measures concerning menus in UD, which capture how one bidder's pick from her menu influences the full allocation and another bidder's menu. We show that (iii) the outcome-effect complexity is $\widetilde{\Omega}(n^2)$, meaning that a bidder can affect the full allocation in a way that is intractable-to-characterize. Finally, we show (iv) the options-effect complexity is $\widetilde{O}(n)$, giving an efficient representation of how one bidder can affect another's menu. Our results for (i) and (iv) stem from a new characterization of how removing one vertex effects the max-weight matching in a bipartite graph. Perhaps interestingly, the complexity in (ii) and (iii) arises not from one bidder's impact on the allocation of items, but their impact on the prices other bidders must pay.

Contents

1 Introduction

Strategyproofness—the classic mechanism-design property saying that straightforward truthful reporting is a dominant strategy—is elegantly embodied by the concept of a player's menu, i.e., the set of outcomes she might receive, holding other players reports fixed. A mechanism is strategyproof if and only if each player receives her favorite item from her menu. In this paper, we investigate in detail the menus of a classic, canonical auction setting: the VCG auction with unit-demand bidders (henceforth, UD). We give a complete characterization of how ascending- and descending-price auctions can describe the menu in UD, and of how different players' menus and the final allocation can affect one another.

Our study is motivated by the observation that, despite the strong theoretical guarantee of strategyproofness, much empirical evidence shows that bidders may still misreport their private value information in strategyproof mechanisms (e.g., [18, 21, 9]). An ample body of work has searched for new market design solutions which might better convey strategyproofness (e.g., [21, 25, 8]). One approach proposed by [14] is to change the way the auction is *described*. In particular, the framework of [14] proposes that the outcome of a (static, sealed-bid auction) should be described to bidder i using an algorithm with the following outline:

(a) Using only the bids other than i's, calculate i's menu $(p_1, \ldots, p_m)$.

(b) Using i's bid v_i, allocate i to their favorite item according to their menu, i.e., $\arg\max_j v_i(j) - p_j$.

(c) Using all bids, calculate the full outcome: all bidders' awarded items and prices charged.

[14] call such an algorithm a *menu-in-outcome* description for bidder i. The premise of [14] is that framing the rules of the auction using a menu-in-outcome description exposes strategyproofness for bidder i. In particular, bidder i can directly see from the above outline that truthful reporting always awards them their favorite item from some menu they cannot influence, giving a one-sentence proof of strategyproofness.

In this paper, we study menu-in-outcome descriptions of UD, along with other complexity notions concerning "picking from a menu." After providing preliminaries in Section 2, in Section 3 we begin by directly studying the structure of a player's menu—the first step of a menu-in-outcome description—and provide a characterization of the menu in UD which underpins our other results. Our characterization specifies how the welfare-maximizing allocation can change when adding an additional bidder, a crucial step towards understanding the relationship between steps (a) and (c) above.[1]

Our first set of main results concerns ascending- and descending-price auctions. These auctions are a widespread, simple, and canonical method of balancing supply and demand, often based on natural price-adjusting processes. For example, in many natural classes of consumer preferences—including UD—these auctions lead to efficient allocations [20, 16, 2]). Thus, ascending- and descending-price auctions are a natural way to describe the outcomes of UD, and an interesting computational model specifically tailored to auction settings. In particular, for our purposes, ascending- and descending-price auctions form a special class of algorithms that take the directly-reported bids as inputs. Our framework allows us to ask whether this natural auction format also

[1]Our characterization in Section 3 implies previous results regarding the VCG prices in UD (i.e., the VCG prices, the essentially-unique prices making UD strategyproof) given by [20, 16]. Unlike these previous results, it also allows us to reason in Section 4 and Section 5 about the relationship between one player's menu and the final, full allocation and prices.

suffices to expose strategyproofness in terms of "picking from a menu," i.e., whether they provide menu-in-outcome descriptions.

In our first set of main results, presented in Section 4, we study strategyproofness-exposing ascending- and descending-price auctions. We demonstrate that ascending-price auctions *can* compute a bidder's menu (i.e., step (a) above), but *cannot* do so while computing the full outcome (i.e., proceeding to step (c) afterwards). In other words, no ascending-price menu-in-outcome description of UD exists. In contrast, we construct a *descending*-price menu-in-outcome description for UD, thereby concurrently exposing strategyproofness for an individual bidder, and describing the outcome for all bidders. This is in perhaps surprising contrast to traditional uses of (dynamic) descending-price auctions, which are often used to implement first-price auctions and are thus not even strategyproof.

In our second set of main results in Section 5, we consider additional models through which the complexity of steps (a)–(c) above can be viewed. We investigate a measure complexity proposed in [15], termed the "outcome-effect complexity". This serves to measure the complexity of how a bidder selecting from her menu in step (a) can affect the overall outcome: it asks the memory complexity of an auction which is unable to query bids in step (c), and must thus know the entire function from i's bid to the outcome. We show that the outcome-effect complexity for UD is $\Omega(n^2)$, as large as possible up to lower-order terms. Hence, varying the report of a single-bidder can affect the outcome in highly non-trivial ways. Interestingly, this high complexity stems from the *prices* in the outcome (and in fact, our menu characterization in Section 3 directly shows that the effect of one bidder on others' *allocation* can be described in a structured, $\widetilde{O}(n)$-bit way). We also investigate the "options-effect complexity" of UD [15], which uses an analogous complexity measure to investigate the relationship between two bidders' menus. In contrast to the high outcome-effect complexity of UD, we show that UD has a *low* options-effect complexity of $\widetilde{O}(n)$, showing that a bidder's choice from her menu can only affect another bidder's menu in a specific, compactly-describable manner.

Our results show that UD is a fairly simple mechanism: bidders' menus are naturally related to each other, and a natural clock auction—perhaps-surprisingly, a descending-price auction—can compute the outcome of UD while exposing strategyproofness via menus.

1.1 Related work

Conceptually, our work sits within the literature on simplicity in mechanism and market design. Much recent work in strategic simplicity was inspired by the obvious strategyproofness framework of [21], which posits changing the way the mechanism interacts with players might make strategyproofness easier to see. Many works apply or refine this framework, including [5, 1, 29, 22, 28, 25, 26], as well as many others. Our work is most closely inspired by [14], who initiate the study of menu descriptions and related outlines for describing mechanisms (Section 4), and to [15], who study menus and their relationship to the mechanism as a whole via different complexity measures (Section 5). These papers focus on matching mechanisms, while we focus on UD. Like [14], and unlike prior work on obvious strategyproofness, we do not actually suggest alternative extensive-form games that can implement different auctions; instead, we study different ways to describe static, direct-revelation auctions (possibly in terms of ascending or descending-price algorithms) such that strategyproofness is exposed. On the empirical side, [9**?**] give evidence that real participants in static mechanisms can better understand strategyproofness under related alternative descriptions / framings. In particular, [9] show the promise of ascending-clock framings, further motivating their study in our paper.

The results of our paper give a strong parallel between UD and the deferred acceptance stable matching mechanism (henceforth, DA), according to the questions investigated by [14, 15].[2] For instance, ascending- and descending-price auctions describing UD are in some ways analogous to applicant- and institution-proposing descriptions of DA studied in [14]. However, there are important differences between the two, both in the source complexity and in our findings. First, in UD, complexity arises from the difficulty of calculating other bidders *prices charged*, where in DA they arise from the difficulty of calculating others' matches. Second, we find that ascending-price menu-in-outcome descriptions for UD are completely impossible (Theorem 4.3), where for DA an applicant-proposing menu-in-outcome description is possible, but it requires memory $\Omega(n^2)$ (as shown by [14]).

Our work also sits within the literature on ascending- and descending-price auctions for UD. Since menus are closely related to Walrasian (i.e., competitive) equilibria, our questions about these auctions are more closely related to classical auction literature (when compared with our questions about complexity measures). [20, 12, 27] establish the fundamental properties of these auctions. [16, 17] study the more general gross substitutes auctions and establish further properties. Papers studying other types of ascending or descending auctions are numerous, and include [23, 10, 24, 4, 2, 3, 13, 6], as well as many others. A related line of work studies notions of communication complexity in these auctions; most relevant is likely [19], who use properties of UD related to Observation 3.8 to bound the complexity of verifying the UD-outcome (according to notions of verification analogous to nondeterministic computation). Also similar in spirit to our work is [7] which explores the computational power of various implementations of "iterative" auctions (e.g., ascending versus descending prices, item versus bundle pricing, and value versus demand queries). Analogous to our results in Section 4, they (among other results) identify classes of valuations which cannot be fully elicited by ascending item prices but can be fully elicited by descending item prices, and vice-versa, demonstrating differences in their computational power.

2 Preliminaries

First, we define auction environments abstractly:

Definition 2.1. *An* auction environment *consists of a set of n* bidders *$\mathcal{B} = \{1, \ldots, n\}$, a set of* items *$\mathcal{I}$, and sets of bidder* types *$\mathcal{T}_1, \ldots, \mathcal{T}_n$. For each i, the elements $v_i \in \mathcal{T}_i$ are valuation functions $v_i : 2^{\mathcal{I}} \to \mathbb{R}_{\geq 0}$, which map subsets of $\mathcal{I}$ to bidder i's utility for receiving that bundle. Outcomes in this environment are allocations $(A_1, \ldots, A_n)$ of items in $\mathcal{I}$ to bidders in $\mathcal{B}$ (i.e., $A_i \subseteq \mathcal{I}$ for every i, and $A_i \cap A_j = \emptyset$ for every $i \neq j$), along with prices $(p_1, \ldots, p_n)$ to be paid by the respective bidders. The utility of player i with type v_i under an outcome with allocation $(A_1, \ldots, A_n)$ and prices $(p_1, \ldots, p_n)$ is $v_i(A_i) - p_i$. A* social choice function *in some auction environment is a function f which maps $\mathcal{T}_1 \times \ldots \times \mathcal{T}_n$ to allocations and prices for each bidder.*

Note that i's preferences over allocations are completely determined by the items allocated to i, along with the price i pays. That is, all outcomes in which i receives the same set of items and

[2]A previous version of [14] (which can be found at `https://arxiv.org/abs/2209.13148v2`) also considered additive and unit-demand auctions, but included only a very simple analysis of descriptions/algorithms that can only query each bidder once, sequentially. In contrast, the present paper considers the much-richer class of descriptions given by ascending- and descending-price auctions, and considers for the first time the complexity measures of [15] in an auction context.

pays the same price are indistinguishable from the point of view of bidder i's utility.

We consider allocation functions that maximize *welfare*, i.e., the sum $\sum_{i\in\mathcal{B}} v_i(A_i)$. Furthermore, we consider the price function that charges the canonical VCG prices. These prices charge bidder i the loss in welfare that the other bidders incur due to the presence of bidder i. That is, when the welfare-maximizing allocation is $(A_1,\ldots,A_n)$, player i is charged price

$$p_i = \left(\max_{A'_{-i}} \sum_{j\neq i} v_j(A'_j)\right) - \sum_{j\neq i} v_j(A_j)$$

(where the max is taken over all allocations A'_{-i} to bidders other than i). It is well-known that the social choice function defined by these allocation and price functions is strategyproof.

Our paper focuses on unit demand valuations, where each bidder only wants one item. Throughout the paper, we write $v_i(j)$ to mean $v_i(\{j\})$.

Definition 2.2. *A unit demand valuation $v_i : 2^{\mathcal{I}} \to \mathbb{R}_{\geq 0}$ is a valuation such that $v_i(S) = \max_{j\in S} v_i(j)$ for each $S \subseteq \mathcal{I}$. We denote the social choice function consisting of the welfare-maximizing allocation, along with the VCG prices, with n unit demand bidders and m items as $\mathsf{UD} = \mathsf{UD}^{m,n}$.*

Our primary objects of study in this auction is a bidder's menu, which are the prices that she would have to pay to win each item, holding all other bids fixed.

Definition 2.3. *Given a profile of valuations $v_{-i} = (v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_n)$, bidder i's* menu, *denote $\mathcal{M}_i^{\mathsf{UD}}(v_{-i})$, is the list of price $(p_j)_{j\in\mathcal{I}}$, where for each j price p_j is the (unique) price such that there exists a v_i such that $\mathsf{UD}(v_i, v_{-i})$ awards player i item j at price p_j.*

In UD, each bidder i receives her favorite item from her menu, i.e. the item j maximizing $v_i(j) - p_j$. This is equivalent to strategyproofness (and this is true for all auctions under a suitable generalization of the definition of menus to bundles instead of just single items), which motivates the detailed study of menus as a lens into strategyproofness.

We make a few assumptions for the sake of concreteness. Let $n = |\mathcal{B}|$ denote the number of bidders. First, we restrict attention to valuation functions and prices that are integers in $\{0,1,\ldots,K\}$, for some K that is polynomial in n, and to the case where $|\mathcal{I}|$ is $O(n)$. (Note though that the scale of these utility values is arbitrary, so 1 unit can represent a single cent or even a smaller unit of currency.) This assumption implies that a single value for a single item, and the identity of a single item, is describable in a small number of bits (namely, $O(\log(n))$-many bits). Under this assumption, $\widetilde{O}(n)$ bits suffices to write down a price for all of the items, facilitating a conceptually clean distinction between $\widetilde{O}(n)$ bits and $\widetilde{\Omega}(n^2)$ bits (which matches the number of bits required to write down all bidders' values for all the items, i.e. the entire input).

Second, we assume that in UD, the allocation rule never awards any bidder more than one item, since doing so can never strictly increase welfare. For this reason, in UD we often refer to allocations as *matchings*, and use notation such as μ for this allocation, and $\mu(b) = x \in \mathcal{I} \cup \{\emptyset\}$ to denote the item which bidder b receives. We refer to the allocation in UD as the max-weight matching.

Third, we assume that UD uses a fixed, deterministic tie-breaking rule to decide between allocations with the same welfare. In fact, we use this tie-breaking rule to assume that the welfare of the max-weight matching is strictly higher than the welfare of every other matching. Since we assume all values are integers, one way to formally model this tie-breaking rule is to add a value of $3^{-(m+\cdot i+j)}$ to bidder i's value for item j, for each $i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$. Under these modified values, the unique max-weight matching is the "lexicographically lowest" max-weight matching under the un-modified values. We assume that UD follows the tie-breaking rule given by these modified values. Thus, for ease of presentation, throughout this paper, we implicitly use these modified values, and in fact make the following assumption:

Assumption 2.4. *In* UD*, no two matchings have exactly the same welfare. In particular, the welfare of the max-weight matching strictly exceeds the welfare of any other matching.*

3 Characterizing the Menu in UD: Removing One Item

Consider a bidder $b \in \mathcal{B}$ in UD, and consider their menu prices according to the VCG pricing rule for some fixed v_{-b}. This price for item $x \in \mathcal{I}$ can be defined via the max-weight matchings under two different instances: (1) v_{-b}, and (2) $(v_b^{(x)}, v_{-b})$ for some $v_b^{(x)}$ which values (only) item x at a huge value (say, twice the welfare of v_{-b}). We refer to the scenario in (2) as bidder b *stealing* item x; in this terminology, the menu price for item x is the total decrease in welfare (of non-b-bidders) when item x is stolen. It turns out that what happens when bidder b steals item x is closely related across all the different items $x \in \mathcal{I}$. In this section, we provide this characterization, which most of our other sections build upon.

3.1 Differences between matchings

We now introduce notation for capturing the difference between two arbitrary matchings. Recall that for unit-demand auctions, we identify the allocation of bidders to items with a matching μ mapping bidders to one (or zero) items. We will use the following definition to capture how stealing a set of item S affects the max-weight matching; we typically apply this definition when $S = \{x\}$ is a singleton, but we define it for a general set S for generality and to aid later discussion (namely, in Section 5.2).[3]

Definition 3.1. *For a matching μ, a* chain *in μ is any sequence $C = \big[(x_0, b_0), (x_1, b_1), \dots, (x_k, b_k)\big]$, such that $x_i \in \mathcal{I} \cup \{\emptyset\}$ and $b_i \in \mathcal{B} \cup \{\emptyset\}$ and such that where $\mu(x_i) = b_i$ for $i \in \{0, \dots, k\}$.*

Consider a set $S \subset \mathcal{I}$, a matching μ of all bidders to items in $\mathcal{I}$ and a matching ν of all bidders to items in $\mathcal{I} \setminus S$. We say that ν differs from μ by S-chains

$$\mathcal{T} = \Big\{ \big[(x_{y,0}, b_{y,0}), (x_{y,1}, b_{y,1}), \dots, (x_{y,k(y)}, b_{y,k(y)})\big] \Big\}_{y \in S}$$

if each $C \in \mathcal{T}$ is a chain in μ, and we have:

- *For each $y \in S$, we have $x_{y,0} = y$.*
- *We have $\nu(b_{y,j}) = x_{y,j+1}$ for each $y \in S$ and each $j \in \{0, \dots, k(y) - 1\}$.*
- *For each $b \notin \{b_{y,j} | y \in S, j \in \{0, \dots, k(y)\}\}$, we have $\mu(b) = \nu(b)$.*

[3]This definition is essentially the same as the notion of an augmenting path from the data structures and max-weight matching literature. To our knowledge, our results concerning the structure of these paths are new.

Say that the weight *of a chain C, denoted $\Delta(C)$, is*

$$\Delta(C) = \sum_{i=0}^{k-1} v_{b_i}(x_{i+1}) - v_{b_i}(x_i).$$

The weight $\Delta(\mathcal{T})$ of $\mathcal{T}$ is the sum of the weights of the chains in $\mathcal{T}$.

In words, ν differs from μ by a set of S-chains if, for each chain $C = [(x_0, b_0), (x_1, b_1), \ldots, (x_k, b_k)]$ and each i, bidder b_i goes from being assigned to item x_i in μ to x_{i+1} in ν, and if the chains collectively describe the entire difference between μ and ν. One can visualize the meaning of a chain as follows: Each pair in C is allocated in μ, but in ν, the item x_0 is no longer allocated to anyone, and the allocation matches the pairs that arise when we move the parentheses one place to the right: $[x_0, (b_0, x_1), (b_1, x_2), \ldots, (b_{k-1}, x_k), b_k]$. In words, the weight $\Delta(C)$ of C is the difference in welfare resulting from reassigning bidders according to C. Note that $\Delta(C)$ will typically be negative, and that b_k does not factor into $\Delta(C)$—in the difference between two matchings, such chains will always have (x_k, b_k) equal to either $(x_k, \emptyset)$ or $(\emptyset, \emptyset)$ (though we sometimes use chains without this property for the sake of proofs). Moreover, note that since we work in a unit-demand setting, if ν differs from μ by $\mathcal{T}$, then no item or bidder can appear in more than one of the chains in $\mathcal{T}$.

Recall that throughout this paper, we assume by Assumption 2.4 there are never any ties between the welfare of any two matchings. We now show that, when μ and ν maximize welfare over their respective set of available items, the set of S-chains by which ν differs from μ is uniquely determined. We defer this proof (and all missing proofs) to the full version of this paper.

Lemma 3.2. *Suppose μ is the welfare-maximizing allocation with all bidders and items $\mathcal{I}$. Let $S \subset \mathcal{I}$ be any set of items, and suppose ν is the welfare-maximizing allocation of items in $\mathcal{I} \setminus S$ to all bidders. Then, there exists a unique set of S-chains $\mathcal{T}$ such that μ differs from ν by T.*

Proof. Fix such a μ and ν, and suppose μ differs from ν by $\mathcal{T}$. We need to show there is a unique chain in $\mathcal{T}$ starting with $(y, \mu(y))$ for each $y \in S$, and that there are no other chains in $\mathcal{T}$. For each $y \in S$, recall that we set $x_{y,0} = y$. To begin, observe that the rest of the chain $\big[(x_{y,0}, b_{y,0}), \ldots, (x_{y,k(y)}, b_{y,k(y)})\big]$ are already uniquely determined, since for each j we set $x_{y,j} = \nu\big(\mu(x_{y,j-1})\big)$ by definition. To finish the proof, we must thus show that no additional chains can appear in $\mathcal{T}$. To see this, suppose for contradiction that $C = [(x_0, b_0), \ldots, (x_k, b_k)] \in \mathcal{T}$ for some $x_0 \notin S$. Then, if $\Delta(C) > 0$, then μ could not possibly be a welfare maximizing allocation of all items, since we could reassign items according to C in μ. On the other hand, if $\Delta(C) < 0$, then ν could not possibly be a welfare maximizing allocation of all items not in S, since $x_0 \notin S$ and we could thus reassign items according to the *reverse* of C in ν, since x_0 can still be allocated in ν.[4] This proves the lemma. □

3.2 Chains from removing one item

We now study the effect on the max-weight matching of removing a single item. Given Lemma 3.2, the following notion is well-defined:

[4]Here, as in the remainder of the paper, we use Assumption 2.4 to (without loss of generality) ignore the possibility that $\Delta(C) = 0$.

Definition 3.3. *For some $y \in \mathcal{I}$, suppose ν is the welfare-optimal matching without y, and μ is the welfare-optimal matching with all items. Define* $\text{Chain}(y)$ *to be the unique $\{y\}$-chain $[(x_0, b_0), (x_1, b_1), \ldots, (x_k, b_k)]$ starting from $y = x_0$ which appears in the difference between ν and μ.*

The defining feature of $C = \text{Chain}(x)$ is that, if μ is the max-weight matching, for any other chain D in μ which starts with $(x, \mu(x))$, we have $\Delta(C) > \Delta(D)$. Note also that, by definition, $-\Delta(\text{Chain}(x))$ equals the menu price of item x, i.e., the price a hypothetical additional bidder would have to pay to receive item x.

The main result of this section is to show that the Chains resulting from removing single items have a simple interrelated structure. First, we prove the main lemma that builds towards this characterization, and says that for any pair of items x_0, x_1, it cannot be the case that both $\text{Chain}(x_0)$ contains x_1 and that $\text{Chain}(x_1)$ contains x_0. (Indeed, the proof of the lemma reveals that if this were the case, then there exists a cycle containing x_0 and x_1 that improves welfare, hence contradicting the welfare-optimality of μ.) Formally:

Lemma 3.4. *If* $\text{Chain}(x_0) = [(x_0, b_0), (x_1, b_1), \ldots, (x_k, b_k)]$ *then* $\text{Chain}(x_1)$ *does not contain* (x_0, b_0).

Proof. Let μ be the max-weight matching with all items, and suppose for the sake of contradiction that $\text{Chain}(x_0) = [(x_0, b_0), (x_1, b_1), \ldots, (x_k, b_k)]$, and $\text{Chain}(x_1) = [(x_1, b_1), (x'_2, b'_2), \ldots, (x'_\ell, b'_\ell)]$ is such that $x_0 = x'_u$ for some u (and thus $b_0 = b'_u$). We break these chains into two components each, as follows:

$$C_0^H = [(x_0, b_0), (x_1, b_1)] \qquad C_1^H = [(x_1, b_1), (x'_2, b'_2), \ldots, (x'_{u-1}, b'_{u-1}), (x_0, b_0)]$$
$$C_0^T = [(x_1, b_1), (x_2, b_2), \ldots, (x_k, b_k)] \qquad C_1^T = [(x_0, b_0), (x'_{u+1}, b'_{u+1}), \ldots, (x'_\ell, b'_\ell)]$$

See Figure 1 for an illustration.

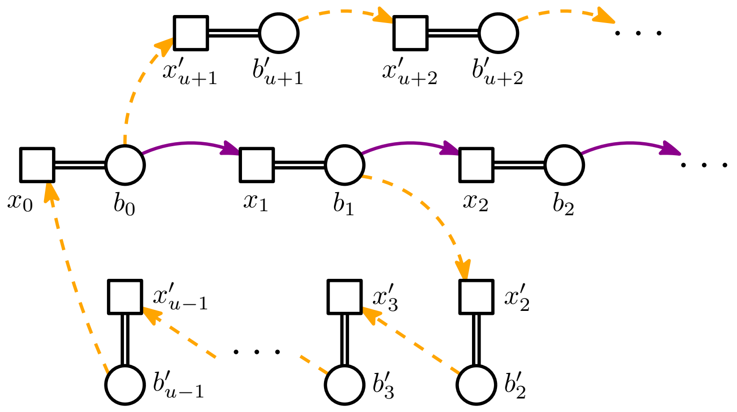

Figure 1: Illustration of the chains we use in the proof of Lemma 3.4.

Notes: Circles represent bidders; squares represent items. The double lines indicate assignments in the max-weight matching with all items. The purple solid arrows indicate re-assignments that are made when x_0 is removed. The orange dashed lines indicate re-assignments that are made when x_1 is removed.

We combine two key properties of these chains to arrive at a contradiction. First, since C_1^T is some chain beginning with (x_0, b_0), we have:

$$\Delta(C_0^H) + \Delta(C_0^T) = \Delta(\text{Chain}(x_0)) > \Delta(C_1^T).$$

Second, dually, since C_0^T is some chain beginning with (x_1, b_1), we have:

$$\Delta(C_1^H) + \Delta(C_1^T) = \Delta(\text{Chain}(x_1)) > \Delta(C_0^T).$$

Summing the above two equations, we get:

$$0 < \Delta(C_0^H) + \Delta(C_1^H) = v_{b_0}(x_1) - v_{b_0}(x_0) + v_{b_1}(x_2') - v_{b_1}(x_1) + v_{b_2'}(x_3') - v_{b_2'}(x_2') + \\ \dots + v_{b'_{u-2}}(x'_{u-1}) - v_{b'_{u-2}}(x'_{u-2}) + v_{b'_{u-1}}(x_0) - v_{b'_{u-1}}(x'_{u-1}).$$

However, this directly contradicts the assumption that μ was the max-weight matching with all items, since items can be reassigned cyclically according to C_0^H and C_1^H as above while increasing the welfare. □

With this lemma in hand, it becomes easy to show that if x_1 is part of the optimal chain for x_0, then the "tail" of Chain(x_0) starting from x_1 must be the optimal chain for x_1. Formally:

Proposition 3.5. *If* Chain$(x_0) = [(x_0, b_0), (x_1, b_1), \dots, (x_k, b_k)]$*, then it must be the case that* Chain$(x_1) = [(x_1, b_1), \dots, (x_k, b_k)]$.

Proof. For the sake of contradiction, suppose Chain$(x_0) = [(x_0, b_0), (x_1, b_1), \dots, (x_k, b_k)]$, and yet

$$\text{Chain}(x_1) = [(x_1, b_1), (x_2', b_2'), \dots, (x_\ell', b_\ell')] \neq [(x_1, b_1), (x_2, b_2), \dots, (x_k, b_k)].$$

By the definition of Chain(x_1), we have

$$\Delta(\text{Chain}(x_1)) > \Delta([(x_1, b_1), (x_2, b_2), \dots, (x_k, b_k)]).$$

By Lemma 3.4, we know that x_0 does not appear in Chain(x_1). However, this means that $C = [(x_0, b_0), (x_1, b_2), (x_2', b_2'), \dots, (x_\ell', b_\ell')]$ is a valid chain of reassignments in μ (i.e., that all items and bidders who appear in C are distinct). However, we have $\Delta(C) > \Delta(\text{Chain}(x_0))$. This contradicts the definition of Chain(x_0), and thus completes the proof. □

3.3 The remove-one graph

We are now ready to give our characterization of the menu in UD. This takes the form of a concise ($\widetilde{O}(n)$-bit) data structure which, simultaneously for each $x \in \mathcal{I}$, represents Chain(x), and thus represents the menu in UD in terms of the losses in different bidders' utilities.

Definition 3.6 (Remove-One Graph)**.** *For a profile of unit demand valuations $\vec{v}$ of bidders other than b, define the* remove-one graph *$G(\vec{v})$ as follows. The vertices of $G(\vec{v})$ are all pairs (x_i, b_j), where x_i is allocated to b_j in the welfare-optimal allocation with all items. Moreover, for each vertex (x_0, b_0), the (directed) graph $G(\vec{v})$ has an edge from (x_0, b_0) to (x_1, b_1), where* Chain$(x_0) = [(x_0, b_0), (x_1, b_1), \dots, (x_k, b_k)]$*. (If item x_0 is unmatched, then the node $(x_0, \emptyset)$ appears in the graph with out-degree 0, and $G(\vec{v})$ also has a vertex $(\emptyset, \emptyset)$ with out-degree 0.)*

If $(x_0, b_0) \to (x_1, b_1)$ is an edge in the remove-one graph, then we say that bidder b_0 *points to* item x_1 in this graph. By definition, there are $O(n)$ vertices in the remove-one graph, and every vertex has out-degree at most 1; hence the graph can be represented in $\widetilde{O}(n)$ bits. Nevertheless, Proposition 3.5 implies that the graph can represent Chain(x) for all items x, and thus the menu prices in UD. Formally:

Theorem 3.7. *Let $G(\vec{v})$ be the remove-one graph. Then, for each item x_0, we have that $\mathrm{Chain}(x_0)$ equals the unique maximum-length chain $C = [(x_0, b_0), (x_1, b_1), \dots, (x_k, b_k)]$ such that for each $i = 0, \dots, k-1$, there is an edge in $G(\vec{v})$ from (x_i, b_i) to (x_{i+1}, b_{i+1}). Thus, for each x_0, graph $G(\vec{v})$ represents the menu price (of a hypothetical additional bidder) as $-\Delta(C)$.*

Proof. For each item x, let $C(x)$ denote the path in $G(\vec{v})$ constructed as in the statement of the theorem. We proceed by induction on the length of $C(x)$. First, if the length of $C(x) = 2$, then $C(x) = \mathrm{Chain}(x)$ directly according to Definition 3.6. Now, suppose that for each $C(x)$ of length at most ℓ, we have $\mathrm{Chain}(x) = C(x)$. Consider some x_0, and let $C(x_0) = [(x_0, b_0), (x_1, b_1), (x_2, b_2) \dots, (x_\ell, b_\ell)]$ have length $\ell + 1$. Observe that $C(x_1) = [(x_1, b_1), (x_2, b_2), \dots, (x_\ell, b_\ell)]$ by the definition of C, and since $\mathrm{Chain}(x_0)$ is of the form $[(x_0, b_0), (x_1, b_1), (x'_2, b'_2), \dots)$ by the definition of $G(\vec{v})$. Thus, Proposition 3.5 implies that $\mathrm{Chain}(x_0) = C(x_0)$, as desired. Thus, by induction, $C(x) = \mathrm{Chain}(x)$ for all items x, which proves the theorem. □

A feature of the remove-one graph that plays an important role in the proofs of later sections is that the remove-one graph and the weights of its constituent chains encode, essentially, multiple related matchings with which the bidders are equally "satisfied". Concretely, we may make the following observation regarding the weights of the chains and the bidder utilities:

Observation 3.8. *Let $G(\vec{v})$ be the remove-one graph. Then for each bidder b_i, consider the item x_i to which she is matched in the optimal matching and the item x'_i which she points to in $G(\vec{v})$. We have that b_i is indifferent between receiving x_i at price $-\Delta(\mathrm{Chain}(x_i))$ and x'_i at price $-\Delta(\mathrm{Chain}(x'_i))$.*

Proof. This follows directly from the definition of the weight of a chain. Observe that

$$-\Delta(\mathrm{Chain}(x_i)) - \big(-\Delta(\mathrm{Chain}(x'_i)\big) = v_{b_i}(x_i) - v_{b_i}(x'_i),$$

which is precisely the difference in price between x_i and x'_i that would make b_i indifferent between the two items. □

All told, our results of this section characterize a bidder's menu in UD in terms of (a compact representation of) the effect of removing any single item.[5]

4 Ascending and Descending Auctions

We now turn to ascending-price and descending-price auctions as computational tools for calculating and describing UD. We begin by giving a constructive, algorithmic definition of an ascending price

[5]If we interchange the roles of bidders and items, which does not change the max-weight matching problem, we can get a representation of the effect of removing any given *bidder*, and hence the VCG prices in a given *outcome* (as opposed to a *menu*) in UD. This remove-one-bidder version of our results builds upon and generalizes various results from the literature. [20] shows that the VCG outcome prices, i.e., the gain in others' welfare from removing one bidder, are a competitive equilibrium. [16] show that this remove-one-bidder characterization extends to the minimum competitive equilibrium under the much larger class of gross substitutes valuations (though beyond UD, these are not the VCG prices). Observation 3.8 is closely related to [19, Lemma 4], which uses Hall's theorem to prove that the prices in UD leave each bidder i indifferent between their outcome in the welfare-optimal allocation and their outcome in the welfare-optimal allocation *excluding some other bidder* j. In contrast to all of these works, our results crucially need to characterize how the max-weight matching *changes* when an item/bidder is removed, which we exploit throughout the paper.

auction.[6]

Definition 4.1. *An* ascending price auction *(with one price per item) is an algorithm for computing the auction in which:*

- *Bidders valuations are only queried through* demand queries*. These take as input a bidder i and prices $p_1, \ldots, p_m$ on each item, and return some bundle of items S maximizing $v_i(S) - \sum_{j \in S} p_j$.*[7]
- *Throughout the run of the auction, the prices used in the demand queries (to any bidder) are* weakly increasing*.*

A descending price auction *is defined analogously, except that throughout the auction, the prices are* weakly decreasing*.*

A clock auction *is either an ascending price auction, or a descending price auction.*

Following [14], we are interested in auctions describing UD while exposing its strategyproofness in terms of picking from a menu. This means we are most interested in *menu-in-outcome descriptions* for some bidder i, which first compute i's menu and i's allocation from the menu, and then proceed to calculate the entire outcome. For completeness, we also compare such descriptions to traditional descriptions (of only the outcome) and to menu descriptions (of only one bidder's outcome).

All told, we study three outlines for describing the auction, i.e. three types of algorithms for UD. First, outcome descriptions, which are simply traditional algorithms that compute the full allocation (and prices) of the auction. Second, menu descriptions for bidder i, which compute i's menu in order to describe i's outcome while exposing strategyproofness. Third, menu-in-outcome descriptions for bidder i, which start from a menu description for player i and then continue on to give an outcome description. In detail, menu-in-outcome descriptions and menu descriptions are defined as follows:

Definition 4.2. *An* menu-in-outcome description *for some mechanism and player i is an algorithm with the following three-step outline:*

(a) Using only $t_{-i} \in \mathcal{T}_{-i}$, the menu of player i with respect to t_{-i} is calculated.

(b) Using $t_i \in \mathcal{T}_i$, player i is guaranteed her favorite i-outcome from her menu.

(c) Using both t_i and t_{-i}, the full outcome $f(t_i, t_{-i})$ is calculated.

A menu description *for player i consists only of steps (a) and (b).*

Note that, in an ascending / descending menu-in-outcome description, the same price trajectories must be maintained across steps (a)-(c); in other words, these price trajectories must remain monotonic across the entire run of the description.

Note that the celebrated CK auction of [11] is an ascending-price outcome description of UD; as we show below, a descending-price menu description of UD follows from results in [27]. In this section, we provide a complete classification of all other types of descriptions of UD in this framework. See Table 1 for an overview.

[6]An alternative "ascending trajectory"-based definition of an ascending-price auction can be found in [16]; we use this price-trajectory viewpoint to establish impossibility results, but use a more algorithmic main definition.

[7]For our model, technically, ties among this value maximization must be broken in some deterministic, consistent way. This detail is immaterial, though.

Table 1: Classification of auctions for UD in our framework.

	Menu Description	Outcome Description	Menu-in-outcome Description
Ascending-price auction	We construct this auction in Section 4.2	The classical CK auction [11]	**Impossible**, as we show in Section 4.1
Descending-price auctions	Follows from results in [27]	(Follows from Section 4.3)	We construct this auction in Section 4.3

4.1 Impossibility result for ascending menu-in-outcome descriptions

To begin, we consider strategyproofness-exposing ascending-price auctions. We show that, despite these auctions serving as a canonical simple way to calculate the full outcome of UD, this auction format is completely unable to calculate one bidder's menu "along the way" to this outcome, i.e., there is no ascending menu-in-outcome description for UD.

Theorem 4.3. *No ascending-price auction can constitute a menu-in-outcome description for* UD.

Proof. Consider the instance in Figure 2. We will show that, when bidders other than b_0 have valuations as in that figure, no ascending auction can constitute a menu-in-outcome description for the additional bidder b_0. First, observe that the menu prices of x_1 and x_2 are both 1 for bidder b_0. As such, an ascending auction must raise the price of both of these items to exactly 1 for the first step of a menu-in-outcome description (i.e., step (1) of Definition 4.2). We now demonstrate that, depending on the type of b_0, the value of both δ_1 and δ_2 must be found by the ascending auction in order to compute the outcomes for all agents. In the case that $v_{b_0}(x_1) = 100$ and $v_{b_0}(x_2) = 0$ we have that the VCG outcome awards x_1 to b_0 at a price of 1 and x_2 to b_2 at a price of δ_1. On the other hand, in the case that $v_{b_0}(x_1) = 0$ and $v_{b_0}(x_2) = 100$ we have that the VCG outcome awards x_2 to b_0 at a price of 1 and x_1 to b_1 at a price of δ_2.

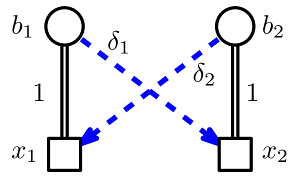

Figure 2: Example showing that it is not possible for an ascending auction to constitute a menu-in-outcome description. Note that δ_1 and δ_2 are in $(0, 0.5)$ above.

Notes: Circles represent bidders; squares represent items. Double edges represent the max weight matching involving only bidders b_1 and b_2 and blue dashed arrows represent the values for b_1 for x_2 and b_2 for x_1.

As a result, it must be that *before* b_0 has chosen her option we must find the values of δ_1 and δ_2. In other words, since δ_1 and δ_2 are less than 1, we must find these values during the ascending-price phase before computing the menu prices of bidder b_0.[8]

[8]This is not true for descending-price auctions, which could first calculate the menu by lowering the price to 1 on each item, and then descend the prices further to find whichever of δ_1 or δ_2 are relevant for the final outcome depending on the choice of b_0. Indeed, as we show below, a descending-price menu-in-outcome description exists.

Observe now that in order to find the value of δ_1 it must be that item x_2 enters the demand set of bidder b_1 at some point in the auction. Suppose this is true at some time t. Then we must have that $p_t(x_2) \leq \delta_1 < 1/2$ and $p_t(x_1) \geq 1 - \delta_1 + p_t(x_2) \geq 1 - \delta_1 > 1/2$. Similarly, to find the value of δ_2 it must be that item x_1 enters the demand set of bidder b_2 at some point in the auction. Suppose that this is true at some time t'. Then we must have that $p_{t'}(x_1) \leq \delta_2 < 1/2$ and $p_{t'}(x_2) \geq 1 - \delta_2 + p_{t'}(x_1) \geq 1 - \delta_2 > 1/2$. But then, we have that $p_{t'}(x_1) < p_t(x_1)$ which implies $t' < t$, and also that $p_t(x_2) < p_{t'}(x_2)$ which implies $t < t'$, a contradiction. We, thus, cannot find the values of *both* δ_1 and δ_2 before computing the menu prices of b_0, and, hence, there exists no ascending auction constituting a menu-in-outcome description. □

4.2 Ascending-price menu description

We have seen that ascending-price auctions for UD cannot compute the outcome while exposing strategyproofness via menus. However, a natural way to describe the auction to bidder i has not yet been ruled out. Namely, can one provide an ascending-price menu description for UD, i.e., a strategyproofness-exposing description of only the outcome of a single player i? In this subsection, we provide such an auction.
Consider the following ascending auction process:

1. Fix some bidder b_i and find the minimal Walrasian equilibrium without b_i using the ascending-price auction of [11] (note that this is exactly the VCG outcome and prices in the market without bidder b_i [11, 16]). Let $\mu(W)$ denote the matched items in the Walrasian equilibrium.

2. Raise the prices of all items in $\mu(W)$ at the same, constant-unit rate until a bidder $b_{i'} = \mu(x_j)$ matched to some item $x_j \in \mu(W)$ would begin to strictly prefer some item $x_{j'} \neq x_j$ over x_j. At this point, "freeze" the price of x_j, remove it from $\mu(W)$, and continue raising the remaining items in $\mu(W)$.

Observation 4.4. *In the second phase of the process bidders matched to an item whose price is still rising can only begin to prefer items whose prices have already been frozen.*

Definition 4.5 (Ascending price graph)**.** *Consider a graph G_i comprising $(n-1)+m$ vertices representing the $n-1$ bidders in $\mathcal{B} \setminus \{i\}$ and the m items in $\mathcal{I}$. We construct edges in G_i by running the ascending price described above, i.e., we initially find a Walrasian equilibrium and then continue raising prices of matched items. We add a directed edge between bidder i' and item j' if bidder i' switches to item j' in the ascending price process from her initially matched item.*

Theorem 4.6. *The ascending price graph G_{b_i} is the remove-one graph $G(\vec{v})$. In other words, the ascending price process above generates the menu for the removed bidder b_i.*

Before proving Theorem 4.6 we first prove a useful auxiliary lemma comparing the prices of an arbitrary item x in the VCG outcome when an arbitrary bidder b_i is removed to the menu price of x for bidder b_i (i.e., $-\Delta(\text{Chain}(x))$ in the remove-one graph of bidder b_i).

Lemma 4.7. *Fix an instance and an arbitrary bidder b_i. Consider the VCG outcome μ and prices on the instance without bidder b_i and let $p^{-i}(x)$ denote the price of item x in this equilibrium. For all $x \in \mathcal{I}$ we have $p^{-i}(x) \leq -\Delta(\text{Chain}(x))$ in the remove-one graph for b_i, i.e., $p^{-i}(x)$ is less than or equal to the menu price of x for bidder b_i.*

Proof. The proof of this lemma follows almost directly from Theorems 4 and 5 of [16]. In particular, Theorem 4 demonstrates that the lowest Walrasian equilibrium prices in a unit-demand economy are exactly the VCG prices. As such, we have that the prices in outcome μ are the minimal Walrasian prices for the economy comprising $\mathcal{I}$ and $\mathcal{B} \setminus \{b_i\}$. On the other hand, Theorem 5 demonstrates that the highest Walrasian equilibrium price in a unit-demand economy for an item $x \in \mathcal{I}$ is the resulting net welfare loss when item x is removed from the economy. This is precisely the menu price of item x for bidder b_i (i.e., the price b_i pays to "steal" item x from the bidders in $\mathcal{B} \setminus \{b_i\}$ changing the remaining economy from comprising $\mathcal{I}$ and $\mathcal{B} \setminus \{b_i\}$ to comprising $\mathcal{I} \setminus \{x\}$ and $\mathcal{B} \setminus \{b_i\}$). □

With Lemma 4.7 in hand we can now complete the proof of Theorem 4.6. We proceed, essentially, by induction on the time step of the auction with Lemma 4.7 acting as a portion of the base case.

Proof of Theorem 4.6. Let $p_t(x)$ denote the price of item x at time step t in the ascending price process described beginning at step 2 (i.e., at time $t = 0$ we have that $p_0(x)$ is the minimal Walrasian price of item x without bidder b_i). Let μ denote the matching of bidders to items at $t = 0$ (i.e., the Walrasian equilibrium allocation of goods when bidder b_i is not present). We proceed via induction on the time step t. Assume that for all $t < t'$ we have the following two properties: (1) For any item x whose corresponding bidder $\mu(x)$ no longer has x in her demand set (i.e., this bidder "switched" away from x) we have that $p_t(x) = -\Delta(\text{Chain}(x))$ (recall by Theorem 3.7 that this is the menu price of x for bidder b_i); and (2) For any item x whose corresponding bidder $\mu(x)$ still has x in her demand set we have that $p_t(x) \le -\Delta(\text{Chain}(x))$.

Note that at $t = 0$ we have that $p_0(x) = 0$ for all unmatched items (by definition of Walrasian equilibrium) and $\mu(x)$ still has x in her demand set for every matched item x so property (1) holds for the base case. To show that property (2) holds at the base case, we may directly apply Lemma 4.7.

We proceed to the inductive step. To see that property (1) is maintained, consider an item x whose matched bidder $\mu(x)$ changes her demand set at time t'. By Observation 4.4 we know that $\mu(x)$ moves to an item x' whose price is no longer rising. But this means that x' left the demand set of $\mu(x')$ in some step $t'' < t'$. As such, by the inductive hypothesis, we know that $p_{t'}(x') = p_{t''}(x') = -\Delta(\text{Chain}(x'))$. Since x' is now in the demand set of $\mu(x)$ we have that $p_{t'}(x) = v_{\mu(x)}(x) - v_{\mu(x)}(x') - \Delta(\text{Chain}(x')) = -\Delta(\text{Chain}(x))$, as desired.

To see that property (2) is maintained, consider the first time t' where there exists some item x with $p_{t'}(x) > -\Delta(\text{Chain}(x))$ and whose corresponding bidder $\mu(x)$ still has x in her demand set. Let x' denote the item to which $\mu(x)$ points to in the remove-one graph. By assumption we have that $p_{t'}(x') \le -\Delta(\text{Chain}(x'))$. We know, however, by Observation 3.8 that $\mu(x)$ is indifferent between receiving x' at price $-\Delta(\text{Chain}(x'))$ and x at price $-\Delta(\text{Chain}(x))$. But this contradicts the fact that $\mu(x)$ still has x in her demand set at time t' since $\mu(x)$ would strictly prefer x' to x at this point. □

4.3 Descending-price menu-in-outcome description

We have seen ascending-price auctions suffice to construct (outcome descriptions and) menu descriptions of UD, but not menu-in-outcome descriptions. Next, we turn to *descending*-price auction, which we recall by Definition 4.1 is one analogous to an ascending-price auction, but where prices monotonically decrease. We show that in a perhaps-surprising contrast with ascending-prices, we can find a descending-price menu-in-outcome description for UD.

Definition 4.8. *An descending price auction (with one price per item) is an algorithm for computing the auction in which:*

- *Bidders valuations are only queried through* demand queries*. These take as input a bidder i and prices $p_1, \dots, p_m$ on each item, and return some*[9] *bundle of items S maximizing $v_i(S) - \sum_{j \in S} p_j$.*
- *Throughout the run of the auction, the prices used in the demand queries (to any bidder) are* weakly decreasing*.*

Consider the following descending price auction process:

1. Excluding bidder b_* run the descending price auction in [27] to find the maximum Walrasian equilibrium prices among bidders in $\mathcal{B} \setminus \{b_*\}$ and items in $\mathcal{I}$. Denote the matching output by this equilibrium as μ. Compute the remove-one graph by asking each bidder b who received a non-empty item $\mu(b) \neq \emptyset$ which other item (including the empty item) $x \neq \mu(b)$ would give them the same utility as $\mu(b)$ at the prices found in the equilibrium.

2. Allow bidder b_* to select her most preferred item i_* (recall that we break ties in an arbitrary, but consistently defined way)[10] at the prices computed by the previous step, match b_* to i_*, all bidders $b \neq b_*$ along Chain(i_*) in the remove-one graph found in step (1) to the item which they point to in Chain(i_*), and all remaining bidders to the items they are matched to in step (1).

3. Lower the prices of all items $\mathcal{I} \setminus \{i_*\}$ uniformly, but stop decreasing the price of item $i \in \mathcal{I} \setminus \{i_*\}$ when $p_i = 0$ or when a bidder $j \neq \mu(i)$ would strictly prefer i over her matched item.

Theorem 4.9. *The above descending auction constitutes a menu-in-outcome description for* UD*.*

Proof. We begin by demonstrating that our descending-price auction process computes the menu of b_* correctly and allocates b_* her preferred b_*-outcome from her menu (i.e., steps (1) and (2) of Definition 4.2). Recall from Theorem 5 of [16] that the highest Walrasian equilibrium price in a unit-demand economy for an item $x \in \mathcal{I}$ is the resulting net welfare loss when item x is removed from the economy. This is the desired menu price of item x for b_* (i.e., the price b_* must pay to "steal" x from the bidders in $\mathcal{B} \setminus \{b_*\}$ changing the remaining economy from comprising $\mathcal{I}$ and $\mathcal{B} \setminus \{b_*\}$ to comprising only $\mathcal{I} \setminus \{x\}$ and $\mathcal{B} \setminus \{b_*\}$). As such, step (1) of our descending-price process computes the menu of b_* correctly (i.e., step (1) of our descending-price process corresponds to step (1) in Definition 4.2). Moreover, by Theorem 3.7 we know that the menu price of b_* for item x is exactly $-\Delta(\text{Chain}(x))$ in the remove-one graph. On the other hand, by Observation 3.8 we know that every bidder $b \neq b_*$ is indifferent between the item she is endowed with and the item she points to in the remove-one graph when price of each item x is $-\Delta(\text{Chain}(x))$. Thus, step (1)

[9]For our model technical, ties among this value maximization must be broken in some deterministic, consistent way. This detail is immaterial, though.

[10]Recall that, by the addition of the small, lexicographic edge weights, we have that all matchings have unique total weight (i.e., Assumption 2.4). We can, in the case of ties, however, ask all bidders to report their *entire* demand set and manually compute the lexicographically first matching among all max-weight matchings.

of the descending-price process also correctly computes the remove-one graph. Since step (2) has b_* select her most preferred item i_* at the menu prices and since the price of i_* is unchanged in step (3), we have that b_* secures her most preferred outcome from her menu (i.e., step (2) of our descending-price process corresponds to step (2) in Definition 4.2).

We now turn to demonstrating that our descending-price auction computes the full outcome (i.e., step (3) of Definition 4.2). In Lemma 4.11 we demonstrate that the outcome reached at the end of step (2) of the descending-price auction process is a Walrasian equilibrium for the *full* economy (including the removed bidder b_*). In step (3), no changes to the allocation occurs, so the allocation of items produced by step (2) is maintained. Moreover, in step (3), we continue to decrease prices for each item x until further doing so would yield $p_x < 0$ or until continuing would cause x to leave the demand set of its optimal bidder. As such, step (3) maintains a Walrasian equilibrium throughout the price decrease process. Since it is known that the set of Walrasian equilibrium prices form a lattice when bidders' valuation functions are gross-substitutes [17] and since for unit-demand valuations we have that the VCG prices are the lowest element in this lattice [16], our auction process correctly computes the VCG outcome in step (3), as desired. □

Observation 4.10. *Consider a bidder b and the item i to which she is matched in step (1) or (2) of the descending-price process above. If b would begin to strictly prefer some item $i' \neq i$ over i during step (3) of the descending-price process, it must be that the price of i' is still decreasing and the price of i is not.*

Lemma 4.11. *Consider running steps (1) and (2) of the above descending-price process and let x_* denote the item chosen in step (2) by the removed bidder b_*. Then, the prices computed by step (1) of the process combined with the allocation which matches: (i) b_* to x_*, (ii) each bidder $b \neq b_*$ contained in* Chain(x_*) *of the remove-one graph to the item she points to in the chain, and (iii) each other bidder to the item she is matched to in step (1); forms a Walrasian equilibrium for the full economy.*

Proof. We first argue that each bidder obtains an element from her demand set. Observe that since the removed bidder b_* chooses her favorite option from the menu in step (2), she clearly gets some element from her demand set at the prices computed in step (1). Moreover, recall that step (1) computes exactly the VCG menu prices of b_* for each item which by Theorem 3.7 is $-\Delta(\text{Chain}(x))$ in the remove-one graph for each $x \in \mathcal{I}$. Recall, though, by Observation 3.8 that at the prices computed in step (1) each bidder $\mathcal{B} \setminus \{b_*\}$ is indifferent between the item she is matched to and the one she points to in the remove-one graph. As such, if we let μ denote the matching output by step (1) and let x_* be the item b_* claims, then the resulting matching obtained by propagating the changes along Chain(x_*) in the remove-one graph (i.e., each bidder along Chain(x_*) receives the item she points to in the remove-one graph and each bidder not along Chain(x_*) receives the item she is matched to in μ) ensures that each bidder obtains an item in her demand set.

We now argue that performing this change yields a welfare-optimal matching for the entire economy. To see this, observe that the welfare produced by any matching is simply the sum of the prices of the sold items and the utilities of the bidders. Recall that step (1) computes a Walrasian equilibrium matching of $\mathcal{I}$ to $\mathcal{B} \setminus \{b_*\}$. As such, all items which are unallocated in step (1) have a price of 0. But then, allowing b_* to claim *any* item x' (not necessarily one she has positive utility for receiving) and propagating along Chain(x') would yield the same revenue as the matching produced in step (1). Moreover, the utility u_b obtained by any bidder $b \neq b_*$ after this change would be the same as it was in the equilibrium found in step (1). We then have that for all $x' \in \mathcal{I}$ the sum of the utilities of all bidders $b \neq b_*$ plus the revenue obtained from sold items by allowing b_* to purchase x' and propagating along $\mathrm{Chain}(x')$ is the same as the welfare produced in step (1). Moreover, by definition of $\mathrm{Chain}(x')$, the resulting matching of items in $\mathcal{I} \setminus \{x'\}$ to bidders in $\mathcal{B} \setminus \{b_*\}$ is welfare-optimal among matchings involving only these bidders and items. As such, the welfare-optimal matching of $\mathcal{I}$ to $\mathcal{B}$ is produced by allowing b_* to claim the item x_* which maximizes her utility which she does in step (2) of the descending-price auction process, as desired. □

5 Complexity Measures

Throughout this paper, we are interested in characterizing how "one bidder picking from their menu" affects the rest of the mechanism. In Section 4, we primarily investigated this topic in the context of ascending- and descending-price menu-in-outcome descriptions. In this section, we consider two complexity measures defined in [15] that investigate this question through a different lens.

First, we consider the complexity of the effect of one bidder's report on the overall outcome.

Definition 5.1. *The* outcome-effect complexity *of a social choice function f is*

$$\log_2 \max_{b \in \mathcal{B}} \left| \left\{ f(\cdot, v_{-b}) \;\middle|\; v_{-b} \in \mathcal{T}_{-b} \right\} \right|,$$

where $f(\cdot, v_{-b}) : \mathcal{T}_b \to A$ is the function mapping each valuation $v_b \in \mathcal{T}_b$ of bidder b to the full outcome $f(v_b, v_{-b})$ (i.e., both the allocation and prices charged).

Briefly, the outcome-effect complexity is motivated by considering the (a)–(c) outline defining a menu-in-outcome description (Definition 4.2), but without the ability to query any bidders' valuations in step (c). Under this restriction, a natural measure of complexity becomes the memory requirement: can the description match (essentially) the memory required to store a single price per item, namely, $\widetilde{O}(n)$-bits, or does the description need to remember essentially the entire valuation of every bidder, namely, $\Omega(n^2)$ bits?

In more detail, and rephrased without reference to menu-in-outcome descriptions, the outcome-effect complexity of a mechanism is the (worse-case) number of bits one might have to remember about v_{-b} in order to calculate the outcome $f(v_b, v_{-b})$ correctly for all possible reports v_b of bidder b. The (log of the) number of functions $f(\cdot, v_{-b})$ exactly captures this complexity, and gives a generic way to measure how complex the effect of one bidder can be. From this viewpoint, characterizing the function $f(\cdot, v_{-b})$, as required to measure the outcome-effect complexity, provides an answer to the question of "how one bidder's pick from their menu determines the overall allocation" which may be even more natural and direct than menu-in-outcome descriptions.[11]

Second, we consider the complexity of the effect of one bidder's report on another bidder's menu.

[11] Note, however, that neither of our results for menu-in-outcome descriptions, nor our results for the outcome-effect complexity, imply the other. On one hand, our results for menu-in-outcome descriptions restrict attention to ascending- and descending-price auctions, but do not restrict the memory used. On the other hand, our results for the outcome-effect complexity concern memory usage, but are not limited to ascending- and descending-price auctions. We examine this connection in more detail in the full version, where we show that an argument similar to the one we use to bound the outcome-effect complexity suffices to get a much weaker form of Theorem 4.3.

Definition 5.2. *The* options-effect complexity *of a social choice function f is*

$$\log_2 \max_{b,c\in\mathcal{B}} \left| \left\{ \mathcal{M}_c^f(\cdot, v_{-\{b,c\}}) \;\middle|\; v_{-\{b,c\}} \in \mathcal{T}_{-\{b,c\}} \right\} \right|,$$

where $\mathcal{M}_c^f(\cdot, v_{-\{b,c\}})$ is the function mapping each valuation $v_b \in \mathcal{T}_c$ of bidder b to the menu $\mathcal{M}_c^f(v_b, v_{-\{b,c\}})$ of c in f.

Like the outcome-effect complexity, this is formalized by the (log of the) number of different functions from bidder b's report. It can also can be motivated via strategyproofness-exposing description, specifically, the question of how two different bidders' menu descriptions interact (i.e., how one bidder's pick from their menu affects the other's menu and vice-versa).

5.1 Outcome-effect complexity & removing one item and one bidder

We now investigate the outcome-effect complexity of UD; say the effect of the report of bidder b_0 on the outcome for all bidders. Since each bidder will end up matched to exactly one item, we can think of this as the complexity of the effect of b_0 "stealing one item". In view of Section 3, which provides a concise characterization of how the maximum weight matching changes when we remove an item, one might conjecture that this effect is "small" (or more precisely, structured). Interestingly, however, this turns out to not be the case.

In contrast to Section 3—which we think of as characterizing the effect of removing one item—the outcome-effect complexity of UD requires studying the effect of removing one item *as well as any one bidder.* To see why, consider the prices charged to bidders other than b_0. When b_0 receives item x, bidder b_i for $i > 0$ is charged a price determined by the welfare that would result if b_i (and b_0's item x) were not included in the market. Thus, informally, we need to know what happens when any item x and any bidder b_i are removed.

We prove that, unlike Section 3, determining these welfare values requires much higher memory complexity: up to lower-order terms, it has the maximum possible memory requirement. This shows that the outcome-effect complexity of UD is large.

Theorem 5.3. *With n bidders and n items, the outcome-effect complexity of UD is $\Omega(n^2)$.*

Proof. Recall that UD must output both a maximum weight matching between bidders and items, as well as the VCG prices charged to each bidder. Let $m = 2k$ for some integer k and consider a market with $m + 1$ bidders and m items. We hold aside one of the bidders and denote her as b_0 and construct a large set of profiles of valuations of the remaining m bidders. We will ultimately show that it takes $\Omega(m^2)$ bits to store the function $\mathsf{UD}(\cdot, v_{-b_0})$ which maps bidder b_0's valuation to the outcome (including prices) for all bidders.

Divide the remaining bidders and the items into two groups, one group comprising "left" bidders $b_1^L, \ldots, b_k^L$ and items $x_1^L, \ldots, x_k^L$ and the other comprising "right" bidders $b_1^R, \ldots, b_k^R$ and items $x_1^R, \ldots, x_k^R$. For each bit vector $\vec{c} = \{c_{i,j}\}_{i,j\in\{1,\ldots,k\}} \subseteq \{0,1\}^{k^2}$, we define a profile of valuations $v_{\vec{c}}$.

In each $v_{\vec{c}}$, the preferences of each $b_1^L, \ldots, b_k^L$ are fixed. Specifically, each such b_i^L values x_i^L at 10 and b_i^L values all other items at 0. Each b_i^R similarly values x_i^R at 10, and each x_j^R at 0 for each $j \neq i$. Now we factor in the bit vector $\vec{c}$: each b_i^R will value some subset of the items x_j^L, as a function of $\vec{c}$. Specifically, b_i^R will value x_j^L at exactly $b_{i,j} \in \{0,1\}$. See Figure 3 for an illustration of these valuations.

The main lemma we need to prove is the following:

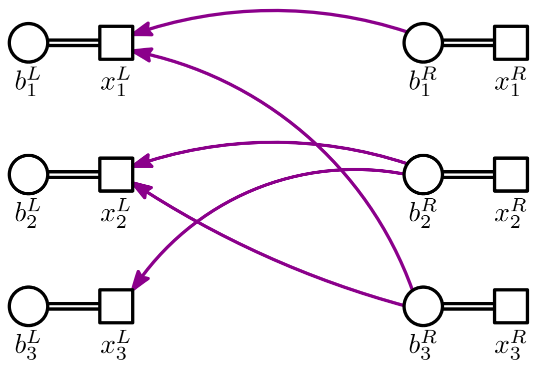

Figure 3: Illustration of the construction in the proof of Theorem 5.3.

Notes: Circles represent bidders; squares represent items. Double edges represent items valued at 10, and purple arrows represent items valued at 1 according to the bit vector $\vec{c}$.

Lemma 5.4. *Let bidder b_0 value item x_i^R at 100 for some $i \in \{1, \dots, k\}$, and value x at 0 for all $x \neq x_i^R$. If bidders other than b_0 have preferences $v_{\vec{c}}$, then in the outcome of* UD*, each bidder b_j^L for $j \in \{1, \dots, k\}$ will pay exactly $c_{i,j} \in \{0, 1\}$.*

To prove this, recall that for each such j, the price bidder b_j^L pays equals the resulting welfare in the optimal matching without b_j^L minus the utility that other bidders actually receive. In the allocation with all bidders, each bidder other than b_0 and b_i^R will match to the (unique) item they value at 10; b_0 will match to x_i^R and b_i^R will go unmatched. In the allocation without b_j^L, each bidder other than b_i^R will match in exactly the same way; the only difference is that there is one more item available to b_i^R, namely, x_j^L. Bidder b_i^R values x_j^L at exactly $c_{i,j} \in \{0, 1\}$. Thus, removing bidder b_j^L would cause the other bidders to gain total utility exactly $c_{i,j}$. This proves Lemma 5.4.

Since each bit vector $\{c_{i,j}\}_{i,j \in [m/2]}$ thus implies a distinct function $\mathsf{UD}(\cdot, v_{\vec{c}})$ from b_0's report to the full outcome, and since there are $2^{\Omega(m^2)}$ such bit vectors, this shows that the outcome-effect complexity of UD is $\Omega(m^2)$. This completes the proof of Theorem 5.3. □

In summary, this result shows that when a new bidder enters the market, despite the fact that by Section 3 she can only have a simple and well-structured effect on the allocation of other bidders, she can have a complex and verbose-to-describe effect on the prices charged to the other bidders.

5.2 Options-effect complexity & removing two items

We now study the options-effect complexity of UD, i.e., the complexity of representing the function $\mathcal{M}_{b_\dagger}(\cdot, v_{-\{b_*, b_\dagger\}})$ from b_*'s report to $b_\dagger$'s menu. In analogy with Section 5.1, where we related outcome-effect complexity to that of removing one item and one bidder, we can relate the options-effect complexity to that of removing two items, as the following proposition shows:

Proposition 5.5. *Consider some profile of valuations $v_{-\{b,c\}}$ of bidders other than b, c. For each pair of items (or the null item) $x, y \in \mathcal{I} \cup \{\emptyset\}$, let $W_{-\{x,y\}}$ denote the maximum achievable welfare among all allocations (to bidders other than b and c) which do not allocate either of the items x, y. Given only access to $W_{-\{x,y\}}$ for each pair $x, y \in \mathcal{I}$, one can determine the function $\mathcal{M}_c(\cdot, v_{-\{b,c\}})$ from b's report to c's menu.*

Proof. We show how to describe $\mathcal{M}_c(\cdot, v_{-\{b,c\}})$ in terms of $W_{-\{x,y\}}$. To this end, consider any type v_b of bidder b, and any item $y \in \mathcal{I}$; we will calculate the price of y on c's menu when other bidders have types $(v_b, v_{-\{b,c\}})$. First, let $W_1 = \max_{x \in \mathcal{I}} v_b(x) + W_{-\{x\}}$; this is the welfare among bidders other than c when c is not included in the market. Second, let $W_2 = \max_{x \in \mathcal{I} \setminus \{y\}} v_b(x) + W_{-\{x,y\}}$;

this is the welfare among bidders other than c when c is matched to item y. Then, by the definition of the VCG prices, c's menu price for y is exactly $W_1 - W_2$. □

For the rest of this section, we focus on characterizing and representing $W_{-\{x,y\}}$, which we term the effect of removing two items. In this language, determining the options-effect complexity resembles the outcome-effect complexity: in Section 5.1, we studied the effect of removing one item and one bidder; now, we must study the effect of removing two items.

Perhaps surprisingly, removing two items is dramatically different from removing one item and one bidder. In particular, we prove that one can represent the effect of removing two items using only $\widetilde{O}(n)$ bits. Briefly, the result will hold since—despite the fact that removing two items is a much more intricate task than removing just one item—we show that there are only two possible items a bidder can move to when two items are removed. In the full version, we discuss and provide some examples of why removing two items is intricate. Nonetheless, we prove that when two items will be removed, each bidder has only two possible items they will be moved to.

5.2.1 Difficulties of reasoning about chains when removing two items

Given Section 3—where we defined $\mathrm{Chain}(x)$ for an item x and used this to characterize the effect of removing one item—a natural approach towards characterizing the effect of removing *two* items is to define the pair of chains that such a removal causes. Namely, recalling Definition 3.1, we make the following definition:

Definition 5.6. *Fix a profile of valuations $\vec{v}$ and let μ be the max-weight matching under $\vec{v}$. For a pair of items $x, y \in \mathcal{I}$, let* $\mathrm{Chain}(x, y) = \mathcal{T}$ *denote the difference between μ, and the max-weight matching that does not allocate items x and y. Then, define* $\mathrm{Chain}(x, y)[x]$ *to be the chain in $\mathcal{T}$ starting with x (and likewise for* $\mathrm{Chain}(x, y)[y]$*).*

In contrast to the clean results about $\mathrm{Chain}(x)$ which we proved in Section 3, in turns out that the chains in $\mathrm{Chain}(x, y)$ for different x, y can have a much more intricate interrelated structure than the chains in $\mathrm{Chain}(x)$ for different x. We give three examples of this in Appendix A. In particular, we show that:

1. It's possible that neither $\mathrm{Chain}(x, y)[x]$ nor $\mathrm{Chain}(x, y)[y]$ coincide with the chains $\mathrm{Chain}(x)$ nor $\mathrm{Chain}(y)$ (Figure 4 on page 24).
2. It's possible that $\mathrm{Chain}(x_0, y)[x_0] = [(x_0, b_0), (x_1, b_1), (x_2, b_2), \ldots]$, yet $\mathrm{Chain}(x_1, y)[x_1] \neq [(x_1, b_1), (x_2, b_2), \ldots]$ (Figure 5 on page 25).[12]
3. It's possible that $\mathrm{Chain}(x)$ is of the form $[\ldots, (x_u, b_u), (x_v, b_v), \ldots]$, yet $\mathrm{Chain}(y, z)[y]$ is of the form $[\ldots, (x_v, b_v), (x_u, b_u), \ldots]$; meaning that removing one item vs. removing two items can reassign items among a pair of bidders in exactly the opposite way (Figure 6 on page 25).

These examples illustrate the comparatively difficult-to-characterize effect of removing two items. Thus, in pursuit of our goal of a concise description of this effect, it seems a different proof approach is needed.

[12]Note that this does not contradict Proposition 3.5, because $\mathrm{Chain}(x_0, y)[x_0]$ and $\mathrm{Chain}(x_1, y)[x_1]$ do not necessarily equal $\mathrm{Chain}(x_0)$ and $\mathrm{Chain}(x_1)$ in a modified profile of valuations where item y is removed. Figure 5 on page 25 also provides an example of this.

5.2.2 Successful construction

We now provide our concise—namely, $\widetilde{O}(n)$-bit—description of the function $S \mapsto W_{-S}$ which maps subsets $S \subseteq \mathcal{I}$ with $|S| \leq 2$ to the maximum welfare W_{-S} achievable with items in $\mathcal{I} \setminus S$. By Proposition 5.5, this will suffice to show that the options-effect complexity of UD is small.

To state this result, we introduce some notation. For the remainder of this section, fix some valuation profile, and the let μ denote the optimal matching among all bidders (excluding b and c whose types are not yet known). For a set of items S, let μ_{-S} denote the max-weight matching which does not allocate items in S.

To begin, consider a bidder b matched to item x in μ. Consider what information about b needs to be recorded to calculate μ_{-S} for any $|S| \leq 2$. Certainly we need to know what b will match to when we remove x; call this item y. Certainly we *also* need to know what b will match to when we remove both x *and* y; call this item z. Our main result of this subsection shows that this is, in fact, *all* one needs in order to calculate $W_{-\{a,b\}}$ for all pairs of items a, b. To make this precise, we first introduce the following definitions:

Definition 5.7 (Remove-Two Valuation Profile)**.** *Given a valuation profile $\vec{v}$, define the* remove-two valuation profile *$\vec{w}$ as follows. For every bidder b, let $\{x, y, z\}$ be the unique triple of items (or $\emptyset$) such that: $\mu(b) = x$, $\mu_{-\{x\}}(b) = y$, and $\mu_{-\{x,y\}}(b) = z$. Now, let $w_b(\alpha) = v_b(\alpha)$ for each $\alpha \in \{x, y, z\}$, and let $w_b(\alpha) = 0$ for all $\alpha \notin \{x, y, z\}$.*

In the remove-two valuation profile, each bidder has at most three items which they value more than 0. Thus, by writing only the names of these items and their bidder's valuation for the item, this valuation profile can be described using only $\widetilde{O}(n)$ bits. We prove that given only the values tracked in the remove-two valuation profile, one can calculate the welfare-optimal matching resulting from removing any two items. These facts together prove the main result of this subsection: a bound on the options-effect complexity of UD. Formally:

Theorem 5.8. *The options-effect complexity of UD is $\widetilde{O}(n)$.*

Proof. As mentioned above, our goal in this proof is to show that there are only three items a bidder might match to when two item are removed. To begin, we build on the facts we know from Section 3 which imply that when one item is removed, there are only *two* possible items a bidder might receive. Fix some valuation profile $\vec{v}$.

More precisely, observe that a direct corollary of Theorem 3.7 is that for each bidder b, there are at most two items that b might match to when one item is removed (namely, this is the item x which b is matched to in the max-weight matching with all items, and the unique item or $\emptyset$ that (x, b) points to in the remove-one graph). Formally:

Corollary 5.9. *For each bidder b, let $\mu(b) = x$, and let $\mu_{-\{x\}}(b) = y$. Then, for every $\alpha \in \mathcal{I}$, we have $\mu_{-\{\alpha\}}(b) \in \{x, y\}$. That is, there are at most two items b might match to in the sub-market where any one item is removed.*

Both parts of the following helpful Lemma 5.10 then follow by applying Corollary 5.9 to the market excluding item u.

Lemma 5.10. *For any bidder b, we have the following:*

1. *For some items $u \neq v$ and $x \neq y$, suppose $\mu_{-\{u\}}(b) = x$ and $\mu_{-\{u,v\}}(b) = y$. Then, for every $w \in \mathcal{I}$, we have $\mu_{-\{u,w\}}(b) \in \{x, y\}$.*

2. *For some items u, v, w which are all distinct, and some items $x \neq y$, suppose $\mu_{-\{u,v\}}(b) = x$ and $\mu_{-\{u,w\}}(b) = y$. Then, for every $t \in \mathcal{I}$, we have $\mu_{-\{u,t\}}(b) \in \{x, y\}$.*

Now, we consider the general case of removing two items, and show that there are only *three* possible items a bidder might receive. This is the main lemma used to prove Theorem 5.8:

Lemma 5.11. *Consider any bidder b. Suppose $\mu(b) = x$, $\mu_{-\{x\}}(b) = y$, and $\mu_{-\{x,y\}}(b) = z$. Now, for any two items $\alpha, \beta \in \mathcal{I}$, we have that $\mu_{-\{\alpha,\beta\}}(b) \in \{x, y, z\}$.*

To prove this lemma, we proceed by using our known results about removing one item to reason about the possible values of $\mu_{-S}(b)$ for S with $|S| = 1$ or with $S \cap \{x, y, z\} = 1$. Namely:

(a) We have $\mu_{-\{\alpha\}}(b) \in \{x, y\}$. This follows from Corollary 5.9.

(b) We have $\mu_{-\{\alpha,y\}}(b) \in \{x, z\}$. Proof: From Theorem 3.7, we know that $\mu_{-\{y\}}(b) = x$. Since $\mu_{-\{x,y\}}(b) = z$ by definition, we thus have that Lemma 5.10, Item 1 implies that $\mu_{-\{\alpha,y\}}(b) \in \{x, z\}$.

(c) We have that $\mu_{-\{\alpha,x\}}(b) \in \{y, z\}$. Proof: By assumption, $\mu_{-\{x\}}(b) = y$ and $\mu_{-\{x,y\}}(b) = z$. Thus, Lemma 5.10, Item 1 implies that $\mu_{-\{\alpha,x\}}(b) \in \{y, z\}$.

Now we examine two cases centered on the interactions of Item a and Item b above. First, suppose we have $\mu_{-\{\alpha\}}(b) \neq \mu_{-\{\alpha,y\}}(b)$. In this case, by Lemma 5.10, Item 1, we have $\mu_{-\{\alpha,\beta\}}(b) \in \{\mu_{-\{\alpha\}}(b), \mu_{-\{\alpha,y\}}(b)\}$. From Item a and Item b above, we have $\{\mu_{-\{\alpha\}}(b), \mu_{-\{\alpha,y\}}(b)\} \subseteq \{x, y, z\}$, and thus $\mu_{-\{\alpha,\beta\}}(b) \in \{x, y, z\}$, which finishes the proof in this case.

Second, suppose that $\mu_{-\{\alpha\}}(b) = \mu_{-\{\alpha,y\}}(b)$. Since we know $\mu_{-\{\alpha\}}(b) \in \{x, y\}$ and that $\mu_{-\{\alpha,y\}}(b) \in \{x, z\}$, this is only possible if $\mu_{-\{\alpha\}}(b) = \mu_{-\{\alpha,y\}}(b) = x$. By Item c, we know $\mu_{-\{\alpha,x\}}(b) \in \{y, z\}$. Thus, we must have $\mu_{-\{\alpha,y\}}(b) \neq \mu_{-\{\alpha,x\}}(b)$. By Lemma 5.10, Item 2, it then follows that $\mu_{-\{\alpha,\beta\}}(b) \in \{\mu_{-\{\alpha,y\}}(b), \mu_{-\{\alpha,x\}}(b)\} \subseteq \{x, y, z\}$. This completes the proof of this case, and of Lemma 5.11.

We now conclude the proof of Theorem 5.8. Let $\vec{w}$ be the remove-two valuation profile. Since $\vec{w}$ differs from the original valuation profile $\vec{v}$ only by setting the value of some items to 0, the welfare of any matching under $\vec{w}$ is at most the corresponding welfare under $\vec{v}$. Moreover, for any items α, β, Lemma 5.11 implies that the max-weight matching $\mu_{-\{\alpha,\beta\}}$ which doesn't allocate α or β has the same weight in $\vec{w}$ as in $\vec{v}$. Thus, the max-weight matching (and welfare) without α or β is the same in $\vec{w}$ and $\vec{v}$.

This proves that using only the remove-two valuation profile, one can calculate W_{-S} for all $|S| \leq 2$. Proposition 5.5 implies that this furthermore suffices to calculate the function from a new bidder's report to the menu. Since it takes $\widetilde{O}(n)$ bits to write down the remove-two valuation profile, this proves Theorem 5.8. □

All told, the outcome-effect complexity of UD is high ($\Omega(n^2)$), and the options-effect complexity is low $\widetilde{O}(n)$. Interestingly, this parallels the finding of [15] for the deferred acceptance stable matching mechanism (DA). However, the reasons for these parallel complexity bounds does not seem clear; for instance, the discussion in the full version of this paper illustrates that our remove-two graph has completely different behavior than the analogous construction for DA (the "unrejection graph") in [15]. We believe that spelling out more precise formal connections between UD and DA is an intriguing direction for future work.

References

[1] Itai Ashlagi and Yannai A. Gonczarowski. 2018. Stable matching mechanisms are not obviously strategy-proof. *Journal of Economic Theory* 177 (2018), 405–425.

[2] Lawrence M Ausubel. 2004. An efficient ascending-bid auction for multiple objects. *American Economic Review* 94, 5 (2004), 1452–1475.

[3] Lawrence M Ausubel and Oleg Baranov. 2017. A practical guide to the combinatorial clock auction.

[4] Lawrence M Ausubel and Paul R Milgrom. 2002. Ascending auctions with package bidding. *The BE Journal of Theoretical Economics* 1, 1 (2002), 20011001.

[5] Sophie Bade and Yannai A. Gonczarowski. 2017. Gibbard-Satterthwaite Success Stories and Obvious Strategyproofness. In *Proceedings of the 18th ACM Conference on Economics and Computation (EC)*. 565.

[6] Eric Balkanski, Pranav Garimidi, Vasilis Gkatzelis, Daniel Schoepflin, and Xizhi Tan. 2022. Deterministic Budget-Feasible Clock Auctions. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2940–2963.

[7] Liad Blumrosen and Noam Nisan. 2005. On the computational power of iterative auctions. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC)*. 29–43.

[8] Inácio Bó and Rustamdjan Hakimov. 2024. Pick-an-object mechanisms. *Management Science* 70, 7 (2024), 4693–4721.

[9] Yves Breitmoser and Sebastian Schweighofer-Kodritsch. 2022. Obviousness Around the Clock. *Experimental Economics* 25 (2022), 483–513.

[10] Giorgos Christodoulou, Vasilis Gkatzelis, and Daniel Schoepflin. 2022. Optimal deterministic clock auctions and beyond. *Leibniz International Proceedings in Informatics, LIPIcs* 215 (2022).

[11] Vincent P Crawford and Elsie Marie Knoer. 1981. Job matching with heterogeneous firms and workers. *Econometrica: Journal of the Econometric Society* (1981), 437–450.

[12] Gabrielle Demange, David Gale, and Marilda Sotomayor. 1986. Multi-Item Auctions. *Journal of Political Economy* 94, 4 (1986), 863–872.

[13] Michal Feldman, Vasilis Gkatzelis, Nick Gravin, and Daniel Schoepflin. 2022. Bayesian and randomized clock auctions. In *Proceedings of the 23rd ACM Conference on Economics and Computation*. 820–845.

[14] Yannai A Gonczarowski, Ori Heffetz, and Clayton Thomas. 2025. Strategyproofness-exposing descriptions of matching mechanisms. (2025). https://arxiv.org/abs/2209.13148 Previous version ("Strategyproofness-exposing mechanism descriptions") appeared as an abstract in the Proceedings of the 24th ACM Conference on Economics and Computation (EC'23).

[15] Yannai A Gonczarowski and Clayton Thomas. 2024. Structural Complexities of Matching Mechanisms. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*. 455–466.

[16] Faruk Gul and Ennio Stacchetti. 1999. Walrasian equilibrium with gross substitutes. *Journal of Economic theory* 87, 1 (1999), 95–124.

[17] Faruk Gul and Ennio Stacchetti. 2000. The English auction with differentiated commodities. *Journal of Economic theory* 92, 1 (2000), 66–95.

[18] John H Kagel and Dan Levin. 1993. Independent private value auctions: Bidder behaviour in first-, second-and third-price auctions with varying numbers of bidders. *Economic Journal* 103, 419 (1993), 868–879.

[19] Sébastien Lahaie and David C Parkes. 2008. On the communication requirements of verifying the VCG outcome. In *Proceedings of the 9th ACM Conference on Electronic Commerce*. 78–87.

[20] Herman B Leonard. 1983. Elicitation of honest preferences for the assignment of individuals to positions. *Journal of political Economy* 91, 3 (1983), 461–479.

[21] Shengwu Li. 2017. Obviously strategy-proof mechanisms. *American Economic Review* 107, 11 (2017), 3257–87.

[22] Pinaki Mandal and Souvik Roy. 2021. Obviously Strategy-proof Implementation of Assignment Rules: a New Characterization. *International Economic Review* 63, 1 (2021), 261–290.

[23] Paul Milgrom and Ilya Segal. 2020. Clock auctions and radio spectrum reallocation. *Journal of Political Economy* 128, 1 (2020), 1–31.

[24] Debasis Mishra and Rahul Garg. 2006. Descending price multi-item auctions. *Journal of Mathematical Economics* 42, 2 (2006), 161–179.

[25] Marek Pycia and Peter Troyan. 2023. A theory of simplicity in games and mechanism design. *Econometrica* (2023). Abstract ("Obvious Dominance and Random Priority") at Proceedings of the 20th ACM Conference on Economics and Computation (EC 2019).

[26] Shiri Ron. 2024. Impossibilities for Obviously Strategy-Proof Mechanisms. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 19–40.

[27] Marilda Sotomayor. 2002. A simultaneous descending bid auction for multiple items and unitary demand. *Revista Brasileira de Economia* 56 (2002), 497–510.

[28] Clayton Thomas. 2021. Classification of Priorities Such That Deferred Acceptance is OSP Implementable. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*. 860.

[29] Peter Troyan. 2019. Obviously Strategy-Proof Implementation Of Top Trading Cycles. *International Economic Review* 60, 3 (2019), 1249–1261.

A Examples of the Effect of Removing Two Items

In this section, we give a few examples illustrating that different effects of removing two items, i.e. Chain$(x, y)[x]$ and Chain$(x, y)[y]$ for different items x and y, can be interrelated in different intricate ways that likely defy a clean characterization like that of the remove-one graph from Section 3. In all the figures of this section: Circles represent bidders. Squares represent items. Doubled edges represent the max-weight matching with no items removed. Solid arrows represent the edges of the remove-one graph. Dashed blue arrows represent additional edges that are in the remove-two graph.

These examples illustrate the following:

1. Figure 4 shows it's possible that neither Chain$(x, y)[x]$ nor Chain$(x, y)[y]$ coincide with the chains Chain(x) nor Chain(y).

2. Figure 5 shows it's possible that Chain$(x_0, y)[x_0] = [(x_0, b_0), (x_1, b_1), (x_2, b_2), \dots]$, yet we also have Chain$(x_1, y)[x_1] \neq [(x_1, b_1), (x_2, b_2), \dots]$.

3. Figure 6 shows it's possible that Chain(x) is of the form $[\dots, (x_u, b_u), (x_v, b_v), \dots]$, yet Chain$(y, z)[y]$ is of the form $[\dots, (x_v, b_v), (x_u, b_u), \dots]$; meaning that removing one item vs. removing two items can reassign items among a pair of bidders in exactly the opposite way.

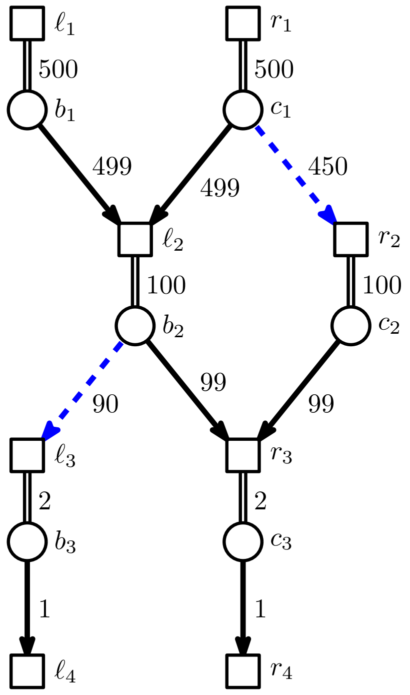

Figure 4: Example illustrating that it is possible that neither Chain$(x, y)[x]$ nor Chain$(x, y)[y]$ coincide with the chains Chain(x) nor Chain(y). One can verify that:

$$\text{Chain}(\ell_1) = [(\ell_1, b_1), (\ell_2, b_2), (r_3, c_3), (r_4, \emptyset)]$$
$$\text{Chain}(r_1) = [(r_1, c_1), (\ell_2, b_2), (r_3, c_3), (r_4, \emptyset)],$$

yet we have

$$\text{Chain}(\ell_1, r_1)[\ell_1] = [(\ell_1, b_1), (\ell_2, b_2), (\ell_3, b_3), (\ell_4, \emptyset)]$$
$$\text{Chain}(\ell_1, r_1)[r_1] = [(r_1, c_1), (r_2, c_2), (r_3, c_3), (r_4, \emptyset)].$$

Intuitively, this construction works by making sure that Chain(ℓ_1) and Chain(r_1) intersect, and by giving two relevant bidders an "outside option". These outside options (namely, the blue dashed edges to r_2 and ℓ_3) do not matter in the remove-one graph, but once ℓ_1 and r_1 are removed simultaneously, Chain$(\ell_1, r_1)[\ell_1]$ and Chain$(\ell_1, r_1)[r_1]$ must each use one of these outside options.

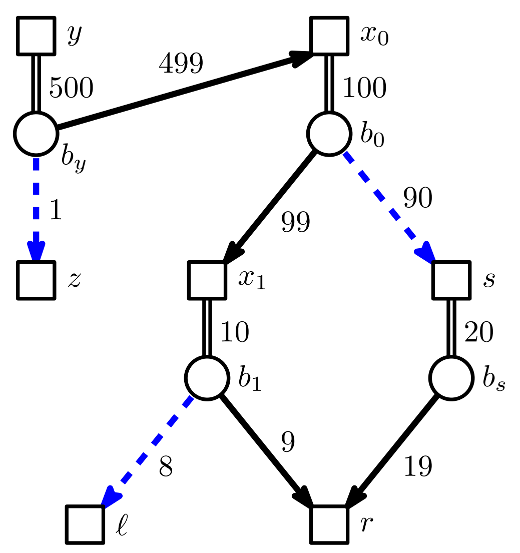

Figure 5: Example illustrating that it is possible that $\text{Chain}(x_0, y)[x_0] = [(x_0, b_0), (x_1, b_1), (x_2, b_2), \ldots]$, yet $\text{Chain}(x_1, y)[x_1] \neq [(x_1, b_1), (x_2, b_2), \ldots]$. One can verify that:

$$\text{Chain}(x_0, y)[x_0] = [(x_0, b_0), (x_1, b_1), (r, \emptyset)]$$
$$\text{Chain}(x_1, y)[x_1] = [(x_1, b_1), (\ell, \emptyset)].$$

Intuitively, this construction works by giving b_y no reasonable option when y and x_0 are removed (thus making $\text{Chain}(x_0, y)[x_0] = \text{Chain}(x_0)$), but making $\text{Chain}(x_1, y)[y]$ involve x_0 and furthermore interfere with $\text{Chain}(x_1)$. We note that, if w denotes the modified valuation profile where item y no longer exists, and if we write $\text{Chain}^{(w)}(\cdot)$ to denote chains under valuation functions w, then

$$\text{Chain}^{(w)}(x_0) = [(x_0, b_y), (z, \emptyset)]$$
$$\text{Chain}^{(w)}(x_1) = [(x_1, b_0), (s, b_s), (r, b_1), (\ell, \emptyset)],$$

so these chains in the modified instance with item y removed do not intersect.

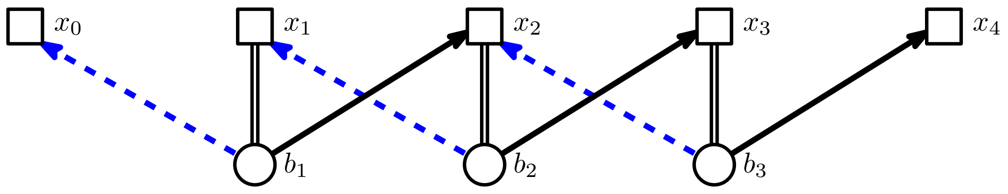

Figure 6: Example illustrating that it is possible that $\text{Chain}(x)$ is of the form $[\ldots, (x_u, b_u), (x_v, b_v), \ldots]$, yet $\text{Chain}(y, z)[y]$ is of the form $[\ldots, (x_v, b_v), (x_u, b_u), \ldots]$. Let each doubled edge represent a value of 100; each solid arrow represent a value of 99; and each dashed blue arrow a value of 90. One can verify that:

$$\text{Chain}(x_1) = [(x_1, b_1), (x_2, b_2), (x_3, b_3), (x_4, \emptyset)]$$
$$\text{Chain}(x_2, x_3)[x_2] = [(x_2, b_2), (x_1, b_1), (x_0, \emptyset)].$$

Intuitively, this construction works by first making it such that the remove-one graph always "moves bidders to the right", but when two items are removed you always "pushes bidders outward" (i.e., away from the two removed items when they are ordered as x_0, x_1, x_2, x_3, x_4) in order to maintain the fact that every bidder receives some item.

B Additional Results on Ascending and Descending Auctions

In this section, we give a remark and an additional result on ascending/descending auctions involving menus for UD. First, we explain the conceptual connection between the outcome-effect complexity of Section 5.1 and the menu-in-outcome descriptions for ascending auctions of Section 4.1. Namely, Section 5.1 can be used to almost directly give a lower bound on the complexity of such an auction. (Note that, as we showed in Section 4.1, something much stronger is true: *no* ascending menu-in-outcome description exists, regardless of complexity. Nevertheless, we include this proof to help illustrate the connections between complexity measures and results for auction descriptions.)

Remark B.1. *Any ascending-price auction based menu-in-outcome description for* UD *requires* $\Omega(n^2)$ *memory.*

Proof. The proof of this fact is, essentially, a corollary of the proof of Theorem 5.3. Recall the construction of Theorem 5.3 involving a set of $m = 2k$ items and $n = m + 1$ bidders. When we hold aside one bidder b_0 and divide the remaining bidders evenly into two groups such that $v_{b_i^L}(x_i^L) = 10$ and $v_{b_i^R}(x_i^R) = 10$ for all $i \in [k]$ observe that this means that in order to compute the menu prices of b_0 via an ascending price trajectory, we must raise the price of each item to 10. On the other hand, recall that the bidders in the right group have $v_{b_i^R}(x_j^L) \in \{0, 1\}$ for each $i, j \in [k]$. Identifying each of these values is necessary for computing the outcomes (which include the prices charged) for the remaining bidders in the left group, but the prices for each of the items in the left group are already at 10 as a result of computing the menu for b_0. As the auction may only ascend prices, there is no way to find these values "on demand" after computing the menu prices for b_0 and these values must then be stored in memory before computing the final menu prices which requires $\Omega(n^2)$ memory. □

Finally, we show that, although the options-effect complexity of UD is low, neither of the ascending-price nor descending-price auction formats can compute the "pairwise menu" of two bidders in UD, which is defined as the set of pairs of prices $\{(p^a_{a,b}, p^b_{a,b})\}_{a,b\in\mathcal{I},a\neq b}$ such that *two* hypothetical additional bidders must pay these prices in order to receive items a and b.[13] This is despite the fact that *both* ascending- and descending-price auctions can compute the menu of any individual bidder.

Theorem B.2. *Neither the ascending-price nor descending-price auction formats can compute the pairwise menu in* UD.

Proof. The proof of this fact utilizes the construction in Figure 7. We first may observe that in order to compute a pairwise menu for two new bidders, it is necessary to discover the values of both δ_1 and δ_2. To see this, first suppose that the two new bidders b_i and b_j have valuations $v_{b_i}(x_1) = 100$ (and v_{b_i} is 0 otherwise) and $v_{b_j}(y_1) = 100$ (and v_{b_j} is 0 otherwise). Then the price that b_i pays is $10 - \delta_1$ and the price b_j pays is $1 - \delta_1$. On the other hand, if they have valuations

[13]Note that the valuations of one of the additional bidders might effect the price the other pays. We show that the impossibility result holds even if we consider the prices charged when the bidders value *only* the items a and b, which is equivalent to specifying that $(p^a_{a,b}, p^b_{a,b})$ are the *minimum* prices required for the additional bidders to get a and b simultaneously. Thus, the same impossibility would hold for more general versions of the definition, e.g., where the pairwise menu is defined as the mapping from the two additional bidders' valuations (v_1, v_2) to the two additional bidders' allocation and prices.

such that $v_{b_i}(x_2) = 100$ (and v_{b_i} is 0 otherwise) and $v_{b_j}(y_2) = 100$ (and v_{b_j} is 0 otherwise) then the price that b_i pays is $10 - \delta_2$ and the price b_j pays is $1 - \delta_2$.

We now argue that it is impossible for either an ascending-price or descending-price auction involving only the two original bidders to compute δ_1 and δ_2. Let b_1 be the bidder with value 10 for x_1 and b_2 be the bidder with value 10 for x_2. In order to learn the value of δ_1 via an ascending/descending process it must be that y_2 is in the demand set of b_1 at some point t in the auction. For this to occur, it must be that $p_t(y_2) \leq \delta_1$ and $p_t(y_1) \geq 1 - \delta_1 + p_t(y_2) \geq 1 - \delta_1$. Analogously, to learn the value of δ_2 it must be that y_1 is in the demand set of b_2 at some point t' in the auction. As such, we have that $p_{t'}(y_1) \leq \delta_2$ and $p_{t'}(y_2) \geq 1 - \delta_2 + p_{t'}(y_1) \geq 1 - \delta_2$. When both δ_1 and δ_2 are less than 0.5 we have that $p_t(y_2) < 0.5$ and $p_t(y_1) > 0.5$ whereas $p_{t'}(y_2) > 0.5$ and $p_{t'}(y_1) < 0.5$. In an ascending-price auction, because $p_t(y_2) < p_{t'}(y_2)$ it must be that $t < t'$ but since $p_t(y_1) > p_{t'}(y_1)$ it also must be that $t > t'$, a contradiction. Likewise, in a descending-price auction, because $p_t(y_2) < p_{t'}(y_2)$ it must be that $t > t'$ but since $p_t(y_1) > p_{t'}(y_1)$ it also must be that $t < t'$, a contradiction. $\square$

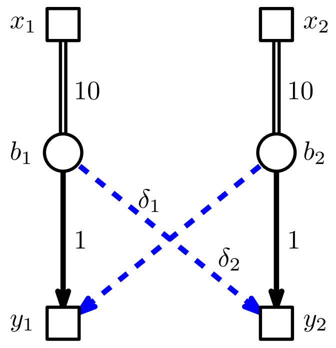

Figure 7: Example showing that it is not possible to compute the pairwise menu using either an ascending or descending auction format when δ_1 and δ_2 are in $(0, 0.5)$ above.

Notes: Circles represent bidders; squares represent items. Double edges represent the max weight matching involving only bidders b_1 and b_2, black arrows represent the items that each bidder would deviate to if her matched item was taken, and blue dashed arrows represent the values for b_1 for y_2 and b_2 for y_1.