The Dissertation Committee for Nevzat Onur Domaniç
certifies that this is the approved version of the following dissertation:

Unit-Demand Auctions:

Fast Algorithms for Special Cases and a Connection to Stable Marriage with Indifferences

Committee:

C. Gregory Plaxton, Supervisor

Vijaya Ramachandran

Eric Price

Evdokia Nikolova

Unit-Demand Auctions: Fast Algorithms for Special Cases and a Connection to Stable Marriage with Indifferences

by

Nevzat Onur Domaniç

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2017

To my parents and my brother, who have been there from day one on this journey; and to my nephew, who joined halfway through and caught up.

Acknowledgments

This dissertation would not have been possible without the guidance, insight, and support of my adviser C. Gregory Plaxton. I would like to express my deepest gratitude for the patient guidance and mentorship he provided during this exciting endeavor. He is not only a source of constant encouragement and inspiration, but also the perfect exemplar of unbounded kindness and positive attitude. I cannot express how grateful and fortunate I am to have had him as my adviser.

I would also like to thank the rest of my dissertation committee, Vijaya Ramachandran, Eric Price, and Evdokia Nikolova, for their valuable comments and feedback.

I am very grateful and lucky to have collaborated with Chi-Kit Lam, I have learned a lot from our discussions and work together.

I would like to thank my fellow graduate students and the CS department staff for their support and friendship. It has been a pleasure to be a member of the graduate student community at UT.

I would like to thank my friends in Austin who made my stay here so much more enjoyable. Special thanks to a few who made Austin a place to call home, they know who they are!

Finally, and most importantly, I would like to thank my mother, Serpil, my father, Yüksel, and my brother, Arman, for their continuous love and support.

Nevzat Onur Domaniç

The University of Texas at Austin
August 2017

Unit-Demand Auctions:

Fast Algorithms for Special Cases and a Connection to Stable Marriage with Indifferences

by

Nevzat Onur Domaniç, Ph.D.
The University of Texas at Austin, 2017
SUPERVISOR: C. Gregory Plaxton

Unit-demand auctions have been heavily studied, in part because this model allows for a mechanism enjoying a remarkably strong combination of game-theoretic properties: efficiency, stability (or envy-freeness), and strategyproofness. One way to derive this mechanism is to specialize the Vickrey-Clarke-Groves (VCG) mechanism to the setting of unit-demand auctions.

In the first part of this dissertation, we focus on developing fast algorithms for implementing the VCG mechanism for compactly representable special cases of unit-demand auctions. We first consider the model with the following special structure: there are n bids, each having two associated real values, a "slope" and an "intercept"; there are m items, each having an associated real value, a "quality"; for each bid u and item v, u offers on v an amount that is equal to the slope of u times the quality of v plus the intercept of u.

Within this model, we present a data structure that processes the bids one-by-one in arbitrary order and maintains an efficient representation of a VCG outcome for the set of processed bids; each bid is processed in $O(\sqrt{m}\log^2 m)$ time. Thus, we solve the problem of computing a VCG outcome of an auction with this form in $O(n\sqrt{m}\log^2 m)$ time. We also present an $O(n \log n)$-time algorithm for computing the VCG prices, given a VCG allocation of an auction with this form.

Next, we study the special case of the aforementioned model where the qualities of the items are evenly-spaced, i.e., the qualities form an arithmetic sequence. This special case is motivated by the following application to the scheduling domain. Consider the problem of scheduling unit jobs on a single machine with a common deadline where some jobs may be rejected. Each job has a weight and a profit, and the objective is to minimize the sum of the weighted completion times of the scheduled jobs plus the sum of the profits of the rejected jobs. It is not hard to see that this problem is equivalent to finding a VCG allocation of an auction with this special form, where we interpret each job as a bid by setting the slope to the negated weight and the intercept to the profit, and we interpret the time slots as items by setting the qualities to 1 through the deadline. We first present an $O(n \log n)$-time algorithm for computing a VCG allocation of an auction with this special form. Then, we describe how to use our algorithm to compute within the same time bound, a VCG allocation of an auction for the case where the item qualities form a nondecreasing sequence that is the concatenation of two arithmetic sequences. This allows us to incorporate weighted tardiness penalties with respect to a common due date into the aforementioned scheduling problem. We also show that certain natural variations of the scheduling problems we study are NP-hard.

In the second part of this dissertation, we explore a connection between unit-demand auctions and the stable marriage model (and more generally, the college admissions model). We present a framework based on two variants of unit-demand auctions: the first variant, a *unit-demand auction with priorities (UAP)*, extends a unit-demand auction by associating a "priority" with each bidder; the second variant, an *iterated UAP (IUAP)*, extends a UAP by associating a sequence of unit-demand bids (instead of a single unit-demand bid) with each bidder. We present a nondeterministic algorithm that generalizes the well-known "deferred acceptance" algorithm for the stable marriage model by iteratively converting an IUAP to a UAP while maintaining a special kind of a maximum-weight matching. Using this framework, we develop a mechanism for the stable marriage model with indifferences that is Pareto-optimal, weakly stable, and group strategyproof for the men. Our results generalize to the college admissions model with indifferences.

Table of Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

In this dissertation, we study matching problems related to two classic game-theoretic market models: the assignment game of Shapley and Shubik [55], a model involving money; and the stable marriage model (and more generally, the college admissions model) of Gale and Shapley [30], a model without money. In the first part of this dissertation, we develop fast algorithms for implementing a well-celebrated mechanism in special cases of the first model, and in the second part of this dissertation, we introduce a variant of that model to solve an open problem related to the second model. We begin with a brief introduction to the field of mechanism design, and to mechanisms with and without money.

The field of *mechanism design* can be broadly described as the science of rule-making. It is the study of designing systems with the goal of implementing a desired *social choice*, i.e., an aggregation of the preferences of participants, or *agents*, toward a single joint decision, where the agents act strategically. The agents are autonomous decision-makers whose objectives are generally different from the designers. For example, a mechanism designer might want to compute a socially efficient allocation of scarce resources or raise significant revenue, while an agent usually only cares about its own utility. Informally, we can think a mechanism as some protocol for interacting with agents, specifying what actions each can take, and mapping those actions into an outcome (including payments for mechanisms with money). One desired property of a mechanism is strategyproofness. A mechanism is *strategyproof* if it is a weakly dominant strategy for any agent to provide truthful preferences, i.e., no individual agent can obtain a better outcome by lying. An even stronger property, namely "group strategyproofness", is the main focus of Chapter 6, and is introduced in Section 1.2.3.

In the domain of mechanism design without money, it is assumed that the agents have preference orderings on the possible outcomes, but the outcomes do not involve the transfer of money. Some classic examples are voting schemes, the house allocation problem [54], and the stable marriage model (and a generalization, the college admissions model) [30]. The main motivation for the second part of this dissertation (Chapters 5 and 6) is the stable marriage model, in which a set of men and women are the agents, where each agent has ordinal preferences over the agents of the opposite sex, and the goal is to find a *matching*, a set of disjoint man-woman pairs, with some desired properties such as being *stable*, i.e., such that no other man-woman pair prefers each other to their partners in the matching.

In a world with money, it is assumed that money can be interpreted to measure how much an agent values an alternative, and can be transferred between agents. The problems we study in the first part of this dissertation (Chapters 2, 3, and 4) assumes *quasilinear preferences*, preferences that have a separable and linear dependence on money: the preference of an agent i is specified by a *valuation function* v_i, $v_i(a)$ denoting the "value" that i assigns to alternative a being chosen; if a is chosen and i pays some money, then the utility of i is equal to $v_i(a)$ minus the payment made by i, and this utility is what i aims to maximize. A mechanism with money not only chooses a social alternative, but also determines the payments made by each agent.

The assignment game of Shapley and Shubik [55] is a classic example of a mechanism design problem with money. The assignment game can be interpreted as a so-called "unit-demand auction", and we adopt this interpretation throughout this dissertation. In a unit-demand auction, we have a set of items up for sale, each with a reserve price set by an associated seller, and there are a number of bidders, each of whom has unit demand (i.e., is interested in purchasing at most one of the items). Each bidder provides a unit-demand bid, which makes a separate monetary offer for each item in some specified subset of the items. If the unit-demand bid is truthful, it specifies the monetary value that the bidder assigns to each of the associated items. Given all of the reserve prices and unit-demand bids, the problem is to determine an appropriate allocation and pricing of the items, where an allocation is required to assign at most one item to any bidder. We assume that if item v is allocated to bidder u at price p in some outcome, and if z is the true value that u places on item v, then u derives a utility of $z - p$ from this outcome. If no item is allocated to bidder u, the utility of u is assumed to be zero.

Unit-demand auctions have been heavily studied, in part because this model allows for a mechanism enjoying a remarkably strong combination of game-theoretic properties: efficiency, stability (or envy-freeness), and strategyproofness. One way to derive this mechanism is to specialize the celebrated Vickrey-Clarke-Groves (VCG) mechanism to the setting of unit-demand auctions. (Clarke [10] and Groves [33] successively generalized Vickrey's second price auction [63] to the general setting with quasilinear utilities and with the natural social choice of maximizing the social welfare and thereby obtained a strategyproof mechanism, that is now called the VCG mechanism.) See Roth and Sotomayor [51, Chapter 8] for proofs of the aforementioned properties (albeit proceeding from first principles instead of using the VCG machinery).

The VCG mechanism guarantees *efficiency* by selecting an allocation that maximizes social welfare. For a unit-demand auction, this means that the allocation is a maximum-weight maximum-cardinality matching of the associated "bid graph", which is a bipartite graph constructed as follows: there is a left node for each bidder (including a dummy bidder for each item, to enforce the reserve price); there is a right node for each item; there is an edge of weight x between bidder u and item v if the unit-demand bid of u includes an offer of x for item v (or if u is the dummy bidder for item v and the reserve price of v is x).

The prices dictated by the VCG mechanism can be characterized in various ways. A desirable property of an outcome (allocation plus pricing) is that it be *stable*, which means that the following conditions hold: the price of any item is at least its reserve price; for any non-dummy bidder u, if we assume that the unit-demand bid provided by u is truthful, then the utility that u derives from the outcome is at least as large as it under any other allocation (with the same prices). (Remark: Sometimes the term "envy-free" is used instead of "stable"; here we use the terminology of Roth and Sotomayor [51, Chapter 8].) It is known that the set of price vectors which form a stable outcome when coupled with an efficient allocation does not depend on the efficient allocation; for this reason, we refer to such price vectors as *stable*. It is also known that the stable price vectors form a nonempty lattice. In this dissertation, we use the characterization that identifies the VCG prices as the (unique) minimum stable price vector [40]. Another characterization of the VCG prices follows directly from the Clarke pivot rule [10]: the price of the item that is assigned to a bidder u is the decrease in the social welfare of the other agents caused by the participation of u, i.e., the total damage that u causes to the other agents. (In economics terminology, the VCG payments make each agent i internalize the externalities that i causes.) VCG prices also correspond to the dual variables computed by the Hungarian method, i.e., they correspond to the prices having the minimum sum among the ones that are the solutions to the dual of the linear program that solves the assignment problem encoding the auction.

Organization. The results of this dissertation can be separated into two parts. In the first part, we focus on developing fast algorithms for implementing the VCG mechanism for compactly representable special cases of unit-demand auctions. We briefly introduce these special cases and state our results in Section 1.1, deferring the formal presentation to Chapters 2, 3, and 4. In the second part, we explore a connection between unit-demand auctions and the stable marriage model (and more generally, the college admissions model). We introduce a framework based on two variants of unit-demand auctions to generalize the well-known "deferred acceptance" algorithm. Using this framework, we develop the first mechanism that enjoys a strong combination of game-theoretic properties, namely strategyproofness and Pareto-stability, for the variant of stable marriage model allowing indifferences (i.e., ties) in the agents' preferences. We introduce the model in Section 1.2, our high-level approach is given in Sections 1.2.1 and 1.2.2, and the formal presentation is deferred to Chapter 5. Then, we show that our mechanism enjoys an even stronger notion of strategyproofness, namely group strategyproofness; our high-level approach is given in Section 1.2.3, and the formal presentation is deferred to Chapter 6. Sections 1.2.1, 1.2.2, and 1.2.3 provide a high-level overview of the main ideas used in the second part of the dissertation; a reader interested in only the formal presentation can skip directly to the relevant chapters (Chapters 5 and 6). Finally, in Chapter 7, we provide some concluding remarks.

1.1 Compactly Representable Unit-Demand Auctions

In the first part of this dissertation, we focus on special classes of unit-demand auctions that can be represented compactly, i.e., classes imposing certain constraints on the space of unit-demand bids so that for each bidder, the unit-demand bid of that bidder can be represented in space less than linear in the number of items. Given the aforementioned desirable properties of the VCG mechanism, our motivation for studying such auctions is to find frameworks to encode unit-demand auctions that are expressive enough to have suitable applications while being restrictive enough to yield efficient algorithms for finding VCG outcomes. For instance, consider a unit-demand auction for last-minute vacation packages in which some trusted third party (e.g., TripAdvisor) assigns a "quality" rating for each package, and each bidder, instead of specifying a separate offer for each package, formulates a unit-demand bid for every package by simply declaring an affine function of the qualities of packages.

The bid graphs corresponding to unit-demand auctions in such a constrained class form a restricted class of bipartite graphs. Both unweighted and weighted matching problems in restricted classes of bipartite graphs have been studied extensively in the literature. Some examples are the convex bipartite graphs, the graphs in which the right vertices can be ordered in such a way that the neighbors of each left vertex are consecutive [29, 31, 36, 41, 62], and two-directional orthogonal ray graphs, which generalize convex bipartite graphs [44]. Our results concerning efficient computation of VCG allocations of compactly representable unit-demand auctions also contribute to this line of research.

1.1.1 A Model with Item Qualities

We now introduce the setting that we study in this dissertation. As in the foregoing scenario about travel-package auctions, we assume that each item has a predetermined quality and that, for each bidder, the offers are given by an affine function of item quality. Thus, each bidder has an offer on each item, so the bipartite graph representing this auction is complete. It is easy to see that such an auction can be represented in space that is linear in the number of bidders and items. More precisely, we consider a unit-demand auction with the following special structure: there are n bidders, each providing a unit-demand bid; each bid has two associated real values, a "slope" and an "intercept"; there are m items, each having an associated real value, a "quality"; for each bid u and item v, u offers on v an amount that is equal to the slope of u times the quality of v plus the intercept of u. We refer to such an actions as a *unit-demand auction with linear edge weights* (*UDALEW*). In the following paragraphs, we state two results regarding UDALEWs.

Our first result, presented in Chapter 2, is a data structure and an algorithm for computing a VCG outcome of a UDALEW. In some of the popular auction sites, e.g., eBay, bidding takes place in multiple rounds. eBay implements a variant of an English auction to sell a single item; the bids are sealed, but the second highest bid (plus a small bid increment), which is the amount that the winner pays, is displayed throughout the auction. We employ a similar approach by accepting the bids one-by-one and by maintaining an efficient representation of a tentative outcome for the growing set of bids. More precisely, we present a data structure that is initialized with the entire set of m items. The bids are introduced one-by-one in arbitrary order. The data structure maintains a compact representation of a VCG outcome (allocation and prices) for the bids introduced so far and for the entire set of items. It takes $O(\sqrt{m}\log^2 m)$ time to introduce a bid, and it takes $O(m)$ time to print the tentative outcome. Thus, we solve the problem of computing a VCG outcome of a UDALEW in $O(n\sqrt{m}\log^2 m)$ time. It is relatively straightforward to process each such bid insertion in $O(m)$ time, yielding an overall $O(nm)$ time bound. Our result is a significant improvement over the latter bound, which is the fastest previous algorithm that we are aware of.

Our second result, presented in Chapter 3, is an $O(n \log n)$-time algorithm for computing the VCG prices of a UDALEW, given a VCG allocation.

1.1.2 Evenly-Spaced Qualities

In Chapter 4, we study two special cases of UDALEWs. In the first special case, we assume that the qualities of the items are evenly-spaced, i.e., the qualities form an arithmetic sequence. Without loss of generality, we can assume that the qualities are the arithmetic sequence 1 through m, because we can represent any arbitrary arithmetic sequence of qualities by appropriately adjusting the intercepts and scaling the slopes. Assuming that $n \geq m$, in Section 4.2, we present an $O(n \log n)$-time algorithm for computing a VCG allocation in this special case. This special case is motivated by the following application to the scheduling domain.

In many scheduling problems, we are given a set of jobs, and our goal is to design a schedule for executing the entire set of jobs that optimizes a particular scheduling criterion. Scheduling with rejection, however, allows some jobs to be rejected, either to meet deadlines or to optimize the scheduling criterion, while possibly incurring penalties for the rejected jobs. Consider the problem of scheduling unit jobs, i.e., jobs with an execution requirement of one time unit, with individual weights (w_i) and profits (e_i) on a single machine with a common deadline ($\overline{d}$) where some jobs may be rejected. If a job is scheduled by the deadline then its completion time is denoted by C_i; otherwise it is considered rejected. Let S denote the set of scheduled jobs and $\overline{S}$ denote the set of rejected jobs. The goal is to minimize the sum of the weighted completion times of the scheduled jobs plus the total profits of the rejected jobs. Hence job profits can be equivalently interpreted as rejection penalties. We represent the problem using the scheduling notation introduced by Graham et al. [32] as:

$$1 \mid p_i = 1, \overline{d}_i = \overline{d} \mid \sum_{S} w_i C_i + \sum_{\overline{S}} e_i \ . \tag{1.1}$$

We assume that the number of jobs is at least $\overline{d}$. It is not hard to see that this problem is equivalent to finding a VCG allocation, which is a maximum-weight matching (MWM) since there are no reserve prices (and hence no dummy bidders), of a UDALEW constructed as follows. Each bid represents a job: the slope of the bid is set to the negated weight of the job, and the intercept of the bid is set to the profit of the job. The $\overline{d}$ items represent the time slots: the qualities of the items are set so that they form the the arithmetic sequence 1 through $\overline{d}$.

The second special case of UDALEWs that we study is the one where the item qualities form a nondecreasing sequence that is the concatenation of two arithmetic sequences. In Section 4.3, we present an $O(n \log n)$-time algorithm for computing a VCG allocation in this special case (we also assume that $n \geq m$). Computing an MWM (a VCG allocation) in this special case allows us to incorporate weighted tardiness penalties with respect to a common due date into Problem 1.1. This more general scheduling problem can be represented using Graham's notation as:

$$1 \mid p_i = 1, d_i = d, \overline{d}_i = \overline{d} \mid \sum_{S} w_i C_i + c \sum_{S} w_i T_i + \sum_{\overline{S}} e_i \ . \tag{1.2}$$

In Problem 1.2, every job also has a common due date d, and completing a job after the due date incurs an additional tardiness penalty that depends on its weight and a positive constant c. The tardiness of a job is defined as $T_i = \max\{0, C_i - d\}$. As in Problem 1.1, we assume that the number of jobs is at least $\overline{d}$.

Shabtay et al. [52] study variants of Problem 1.1 that consider splitting the scheduling objective into two criteria: the scheduling cost, which depends on the completion times of the jobs, and the rejection cost, which is the sum of the penalties paid for the rejected jobs. In addition to optimizing the sum of these two criteria, the authors study other problems such as optimizing one criterion while constraining the other, or identifying all Pareto-optimal solutions for the two criteria. The scheduling cost in that work is not exactly the weighted sum of the completion times, but several other similar objectives are considered. In Section 4.4, we show via reductions from the partition problem that, if we split the scheduling objective of Problem 1.1 in the same manner into two criteria, then the problems of optimizing one of this two criteria while bounding the other are NP-hard.

1.2 Stable Marriage with Indifferences

In the second part of this dissertation, we explore a connection between unit-demand auctions and the stable marriage model (and more generally, the college admissions model) of Gale and Shapley [30]. We introduce a framework based on two variants of unit-demand auctions to generalize the well-known "deferred acceptance" algorithm. Using this framework, we develop the first mechanism that enjoys a strong combination of game-theoretic properties, namely strategyproofness and Pareto-stability, for the variant of stable marriage model allowing indifferences in the agents' preferences (also known as stable marriage with weak preferences). Then, we show that our mechanism also enjoys an even stronger notion of strategyproofness, namely group strategyproofness. We start with an introduction to the stable marriage model, the deferred acceptance algorithm, and the game-theoretic properties that we seek; we defer the definition of group strategyproofness to Section 1.2.3. We present the high-level approach of our mechanism in Section 1.2.1, and we introduce the two variants of unit-demand auctions that are instrumental in our results in Section 1.2.2.

In the most basic form of the stable marriage model, we are given n men and n women, each having strict ordinal preferences over the members of the opposite sex. We assume that the preferences are "complete", i.e., each agent has preferences over all n members of the opposite sex. The objective is to find a matching (i.e., a collection of disjoint man-woman pairs) M of cardinality n that does not admit a *blocking pair*, that is, a man i and a woman j such that i prefers j to his match in M and j prefers i to her match in M. Such a matching is said to be *stable*. Gale and Shapley [30] establish a number of fundamental properties of stable marriage by reasoning about an elegant iterative algorithm known as the *deferred acceptance (DA) algorithm*. The DA algorithm repeatedly updates a tentative matching until a final matching is obtained. The initial tentative matching is the empty matching. In each iteration, an arbitrary unmatched man i is chosen, and i "proposes" to the highest-ranked woman j on his preference list to whom i has not yet proposed. If woman j is unmatched, she tentatively accepts i's proposal. If woman j is tentatively matched to another man i', then j consults her preference ranking to determine whether to reject i's proposal (and continue to be matched with

i'), or to reject i' and tentatively accept i's proposal.

The DA algorithm is nondeterministic since there can be more than one unmatched man at the start of an iteration. The algorithm terminates when there are no unmatched men (and hence also no unmatched women). It is easy to argue that this happens within n^2 iterations, so the algorithm runs in polynomial time. Moreover, the output matching is easily shown to be stable. It is also not difficult to prove that this nondeterministic algorithm is *confluent*, that is, the final matching produced by the algorithm does not depend on the nondeterministic choices made by the algorithm. This shows that the DA algorithm defines a (deterministic) mechanism, which we refer to as the *DA mechanism*. Gale and Shapley [30] prove that the stable matching M produced by the DA mechanism is *man-optimal*: for any man i and any stable matching M', i weakly prefers his match in M to his match in M'. Gale and Shapley [30] also prove that the stable matching M produced by the DA mechanism is *woman-pessimal*: for any woman j and any stable matching M', j weakly prefers her match in M' to her match in M. Symmetrically, the woman-proposing version of the DA mechanism produces a stable matching that is woman-optimal and man-pessimal.

Roth [48] shows that, even for the simplest form of the stable marriage model, no stable matching mechanism is strategyproof for *all* agents. On the other hand, Roth [48] proved that the man-proposing (resp., woman-proposing) DA mechanism is strategyproof for the men (resp., for the women). Throughout the remainder of this dissertation, when we say that a mechanism for a stable marriage model is strategyproof, we mean that it is strategyproof for the agents on one side of the market; moreover, unless otherwise specified, it is to be understood that the mechanism is strategyproof for the men.

Gale and Shapley [30] also consider a generalization of the stable marriage model which they call the college admissions model. In this generalization, we have students and colleges instead of men and women, students have preferences over the colleges, colleges have preferences over students and also over groups of students[1], each student seeks to obtain a single slot at one college, and each college has a specified number of slots available. The preferences of an agent are allowed to be "incomplete" — that is, an agent is allowed to categorize certain agents on the other side as unacceptable. This type of two-sided matching model has been used for decades to assign residents to hospitals, and more recently, for a wide variety of other applications [42, Section 1.3.7]. Most of the properties mentioned above for the stable marriage model carry over to the college admissions model. For example, the DA mechanism is easily adapted to the college admissions setting and the student-proposing version is student-optimal and strategyproof for the students [49] (assuming the college preferences are "responsive"). Some properties mentioned above for the stable marriage model do not carry over to the college admissions model. For example, the college-proposing DA mechanism is not strategyproof for the colleges [49]; the proof makes use of the fact that the

[1]In this dissertation, we assume some form of consistency between the preferences of a college over individual students and over groups of students. See the definition of responsive preferences in Section 5.5.

colleges do not (in general) have unit demand (i.e., a college may have more than one slot to fill). Throughout the remainder of this dissertation, when we say that a mechanism for a college admissions model is strategyproof, we mean that it is strategyproof for the students.

In real-world applications of these models, such as school choice [2, 25, 26], indifferences (i.e., ties) in the preferences of agents arise naturally and render the problems considerably more complex. In order to simplify the high-level presentation in this chapter, we focus on the special case of stable marriage with complete and weak preferences (SMCW), it is easy to adapt these results to the stable marriage model with incomplete and weak preferences (SMIW) (see Section 5.4 for related definitions). With suitable restrictions on the preferences of the colleges over groups of students, the results discussed in this section all generalize to the college admissions model with weak (and possibly incomplete) preferences (CAW) (see Section 5.5 for related definitions and results).

Three main stability notions are considered in the literature when weak preferences are allowed (see, e.g., Manlove [42, Chapter 3]): weak stability, strong stability, and super-stability. Here we introduce the first notion of stability; we touch on the second one in Chapter 7. We say that a man i and a woman j form a *strongly blocking pair* with respect to a matching M if (1) i prefers j to his match in M and (2) j prefers i to her match in M. A matching that does not admit a strongly blocking pair is said to be *weakly stable*. In this dissertation, we focus on weak stability because every SMCW instance admits a weakly stable matching, but not necessarily a strongly stable or a super-stable matching. One way to obtain a weakly stable matching is to break ties arbitrarily and apply the DA mechanism to the resulting stable marriage instance with strict preferences.

A matching is said to be *Pareto-optimal* if there is no other matching that is strictly preferred by at least one agent and weakly preferred by all agents. Sotomayor [61] argues that a natural solution concept for two-sided matching problems with indifferences is given by the set of matchings that are Pareto-optimal and weakly stable, and refers to the matchings in this set as *Pareto-stable matchings.* Sotomayor observes that the set of Pareto-stable matchings is guaranteed to be nonempty because we can start from an arbitrary weakly stable matching and perform a finite sequence of Pareto improvements. This procedure works because any matching M' that is obtained by applying a Pareto improvement to a weakly stable matching M is itself weakly stable; to see this, observe that if M' admits a strongly blocking pair (i, j), then (i, j) also strongly blocks M, a contradiction. Based on this procedure, Erdil and Ergin [26] present a polynomial-time algorithm for computing Pareto-stable matchings in a special case of the CAW model (see Section 5.1 for the description of the model). The algorithm of Erdil and Ergin does not provide a strategyproof mechanism.

1.2.1 A Strategyproof Pareto-Stable Mechanism

In Chapter 5, we provide the first mechanism for the stable marriage problem with indifferences that is Pareto-stable and strategyproof for the men. This mechanism is developed using a framework based on two variants of unit-demand auctions, which are presented in Sections 5.2 and 5.3. Section 5.5 generalizes our mechanism to the college admissions model assuming that the preferences of the colleges are "minimally responsive", a notion to be defined in that section. We can also handle the class of college preferences "induced by additive utility", a notion to be defined in Section 5.5.2.

In this section, we give an overview of our approach and the motivation behind our framework based on the two variants of unit-demand auctions. At the highest level, our approach is to identify a suitable generalization of the DA mechanism. One natural idea that we use is that when an unmatched man is chosen to "propose", he proposes to all of the women in his next tier of preference (i.e., the highest tier of women to whom he has not yet proposed). Somewhat less clear is how the women should respond to such a multi-pronged proposal. Our high-level approach involves maintaining a "revealed graph" representing all of the proposals that have been revealed so far. This is a bipartite graph encoding a variant of a unit-demand auction where the bidders correspond to the "revealed" tiers of men's preferences and the items correspond to the women. An offer for the item corresponding to a woman j is included in the unit-demand bid of the bidder corresponding to a man-tier τ of a man i if i has proposed to j (i.e., j is a prong in some multi-pronged proposal made by i) and j is in τ.

We use the revealed graph to guide the update of the current matching in response to a new multi-pronged proposal by an unmatched man: we maintain the invariant that the current matching is a maximum-weight matching (MWM) of the current revealed graph, where the edge weights (the amounts offered by unit-demand bids) are determined as follows. We first transform the women's weak preferences over the men into real-valued preferences over the men, i.e., each woman assign a real number to each man, in a manner that is consistent with the woman's weak preferences: if woman j prefers man i to man i', then the value j assigns to i is greater than the value she assigns to i'; if j is indifferent between i and i', then j assigns the same value to i and i'. Then the amount of the offer representing a revealed proposal from a man i to a woman j is set to the value that j assigns to i.

When we discussed the key properties of the DA algorithm in Section 1.2, we mentioned that it is confluent: even though the algorithm nondeterministically chooses an unmatched man to propose in each iteration, the output of the algorithm is uniquely determined (i.e., the man-optimal stable matching). To achieve strategyproofness, we find it convenient to ensure that our generalized DA mechanism enjoys a similar confluence property: even though the algorithm nondeterministically chooses an unmatched man to propose in each iteration, the set of men who are matched in the final matching is uniquely determined. (In the event that multiple MWMs match the same set of men, we allow our algorithm to output any such MWM as the final matching, so the output matching is not uniquely determined.)

To complete this high-level description of our generalized DA algorithm, we describe how we refine the update step in order to enforce the confluence property stated in the previous paragraph. Consider an update step, and let $\mathcal{M}$ denote the set of MWMs of the revealed graph. We plan to pick one of the MWMs in $\mathcal{M}$ as the new tentative matching. If all MWMs in $\mathcal{M}$ match the same set of men, then it does not matter which one we pick, informally because the resulting set of single men will be the same, and hence the set of men who are available to propose at the next iteration will be the same. On the other hand, if different MWMs in $\mathcal{M}$ match different set of men, then choosing between them can lead to different sets of unmatched men at the next iteration, and thereby to different revealed graphs at the end of the algorithm, breaking the desired confluence property. To prevent this, we define a class of "greedy" MWMs, and we maintain the invariant that the current matching is a greedy MWM of the revealed graph.

In the next section, we first briefly introduce the notion of a "unit-demand auction with priorities" (UAP) that allows us to define the greedy MWMs alluded to in the preceding paragraph by extending the notion of a unit-demand auction through the introduction of a "priority" with each bidder. UAPs play the role of representing the revealed graphs mentioned above in the high-level description of our mechanism. Then, building on the UAP notion, we introduce the notion of an "iterated UAP" (IUAP) and state a number of important properties of IUAPs; these properties are nontrivial to prove, and provide the technical foundation for our main results. An IUAP allows the bidders, called "multibidders" in this context, to have a sequence of unit-demand bids instead of a single unit-demand bid; each unit-demand bid in such a sequence represents a tier in the preferences of the man associated with the multibidder. At the core of our strategyproof Pareto-stable mechanism is a nondeterministic algorithm that implements a certain mapping from an IUAP to a UAP and produces a greedy MWM of the resulting UAP, which can be interpreted as a mechanism for IUAPs; this mapping formalizes and encompasses the confluence property alluded to in the preceding two paragraphs, which is built upon the results regarding greedy MWMs in UAPs. At each iteration, the algorithm nondeterministically chooses an unmatched multibidder and this multibidder reveals its next unit-demand bid, analogous to the DA algorithm in which a nondeterministically chosen unmatched man reveals his next choice at each iteration. The algorithm, and our whole mechanism, can be implemented in $O(n^4)$ time by utilizing a suitably modified Hungarian iteration to update the greedy MWM when a new unit-demand is revealed, as described in Sections 5.2.3 and 5.3.2.

1.2.2 Iterated Unit-Demand Auctions with Priorities

As briefly discussed in the previous section, the framework we use to develop our strategyproof and Pareto-stable mechanism is based on two variants of unit-demand auctions, one of which is built on the other. The first variant, a *unit-demand auction with priorities (UAP)*, extends a unit-demand auction by associating a “priority” with each bidder (see Section 5.2). Let U denote the set of bidders in a UAP, and let $\mathcal{I}$ denote the set of all subsets U' of U such that some MWM of the bid graph associated with the UAP (some VCG allocation of the UAP) matches all of the bidders in U'. It is straightforward to prove that $(U, \mathcal{I})$ is a matroid (Lemma 5.2.1 in Section 5.2.1). It follows that the matroid greedy algorithm can be used to determine an MWM of the bid graph (associated with the UAP) such that the sum of the priorities of the matched bidders is maximized. We call such MWMs *greedy*. Due to the matroid structure associated with the MWMs, a standard matroid result that follows easily from the exchange property and the correctness of the matroid greedy algorithm implies that all the greedy MWMs of a UAP have the same distribution of priorities (Lemma 5.2.2 in Section 5.2.1). In what follows, we refer to the variant of the VCG mechanism for unit-demand auctions that produces an arbitrary greedy MWM as the allocation (as opposed to an arbitrary MWM) as the *UAP mechanism*.

The second variant, an *iterated UAP (IUAP)*, extends a UAP by associating a sequence of unit-demand bids (instead of a single unit-demand bid) with each bidder, called “multibidders” in this context (see Section 5.3). In an IUAP, we require each multibidder priority to be a unique value which the multibidder does not get to choose, like a social security number. The high-level idea is that a multibidder starts out using the first unit-demand bid in their sequence, only moving on to the second one once the first has been rejected, and to the third once the second has been rejected, and so on. To make this idea precise, we need to specify how to determine when a given unit-demand bid has been rejected. One way to do this is to start out with the UAP in which each multibidder uses the first unit-demand bid in their sequence. By applying the UAP mechanism to this UAP, we obtain a unique set of matched multibidders (due to the result stated in the preceding paragraph and since we require all multibidder priorities to be distinct), and hence also a unique set of rejected multibidders. (This is analogous to the fact that if all of the edge weights in a connected, undirected graph are distinct, then the graph has a unique minimum-weight spanning tree.) We then update the UAP by having the rejected multibidders move on to their second unit-demand bid (if any), and by once again applying the UAP mechanism to determine the next set of rejected multibidders. Continuing in this manner, we eventually arrive at a UAP where each rejected multibidder has exhausted their full sequence of unit-demand bids, at which point we return the output of the UAP mechanism on the current UAP as the final output of the IUAP mechanism.

In the foregoing description of the IUAP mechanism, several rejected multibidders might "reveal" a new unit-demand bid at a given iteration. We find the following natural variation (Algorithm 5.1 in Section 5.3.1), which is more akin to the DA mechanism, to be more useful for the purposes of analysis: at each iteration, choose (nondeterministically) an unmatched multibidder who has not exhausted their entire sequence of unit-demand bids, and update the current UAP by having this multibidder reveal their next unit-demand bid. As in the case of the DA mechanism, where we start out with the empty matching, in this variation it is desirable to start out with no unit-demand bids revealed. A crucial property of this nondeterministic mechanism is that it is confluent (Lemma 5.3.4 in Section 5.3.1): All executions result in the same final UAP as that produced by the IUAP mechanism of the preceding paragraph. Confluence plays a central role in allowing us to prove various important structural properties of IUAPs. For example, confluence enables us, for the purpose of analysis, to delay introduction of any unit-demand bids of a particular multibidder until that multibidder is the sole unmatched multibidder whose sequence of unit-demand bids has not been exhausted. In a key technical lemma (Lemma 5.3.8 in Section 5.3.3), we exploit this technique to show that the remaining multibidders (i.e., the multibidders other than the one we delayed) determine a "threshold" price for the delayed multibidder for each item, thereby allowing us to easily characterize the utility that the delayed multibidder would derive from any sequence of unit-demand bids they might choose to submit.

Our strategyproof Pareto-stable mechanism for the SMIW model corresponds to the DA-like variation of the IUAP mechanism described in the preceding paragraph. Each man (resp., woman) in the SMIW instance corresponds to a multibidder (resp., item) in the IUAP instance. The sequence of unit-demand bids associated with a multibidder t corresponding to a man i is determined as follows: the kth unit-demand bid of t includes an offer for each item corresponding to a woman in the kth tier of preference of i; the amount offered for an item v corresponding to a woman j is chosen to be the value that the valuation profile of woman j assigns to man i as described in the previous section. Because of this correspondence, it turns out to be straightforward to use Lemma 5.3.8 regarding the threshold prices mentioned at the end of the preceding paragraph to establish that our SMIW mechanism is strategyproof for the men. Other properties that we establish for IUAPs allow us to show that the matching produced by this mechanism is Pareto-optimal and weakly stable.

1.2.3 Group Strategyproofness

A stronger notion of strategyproofness is *group strategyproofness*: A mechanism is group strategyproof if no group of agents can misreport their preferences in such a way that all of the group members are better off.[2] Recall that for the two-sided matching problems that we consider, it is known that no stable matching mechanism is strategyproof for *all* agents [48]. With the restriction that all agents' preference are strict, Roth [48] proved that the DA mechanism is strategyproof for the men, and independently, Dubins and Freedman [20] showed that the DA mechanism is group strategyproof for the men. (See Roth and Sotomayor [51, Chapter 4] for a somewhat simpler proof of the Dubins and Freedman [20] result.) We remark that the notion of group strategyproofness studied in this dissertation assumes no side payments within the coalition of men. It is known that group strategyproofness for the men

[2]A mechanism is *strongly* group strategyproof if no group of agents can misreport their preferences in such a way that all of the group members are at least as well off, and some group member is better off. It is known that strong group strategyproofness for the men is impossible for the stable marriage model with strict preferences [51, Chapter 4].

is impossible even for the stable marriage model with strict preferences when side payments are allowed [51, Chapter 4].

On the basis of the result that the DA mechanism for the stable marriage model with strict preferences is group strategyproof for the men, and since our mechanism is a generalization of the DA mechanism that allows for indifferences in preferences, it is natural to hope that our mechanism is also group strategyproof for the men. In Chapter 6 we show that this is the case; we prove that the mechanism in question (the mechanism introduced in this dissertation) coincides with a more recent group strategyproof Pareto-stable mechanism (for the same model) introduced by Domaniç et al. [17].

Group Strategyproofness via the Generalized Assignment Game

The techniques used to develop the group strategyproof Pareto-stable mechanism of Domaniç et al. [17] are different than the ones used in this dissertation; instead of generalizing the deferred acceptance mechanism by employing variants of unit-demand auctions, we cast the stable marriage problem as an appropriate market in the model of Demange and Gale [14], and compute a man-optimal outcome within that model. Demange and Gale's model generalizes the assignment game model of Shapley and Shubik [55] to allow agents to have arbitrary, but continuous, invertible, and increasing utility functions. In Demange and Gale's model, the agents can express more complex utilities than in the assignment game, e.g., in order to acquire an item with a price higher than a certain budget, the agent may decide to take out a loan, which could cause the utility to drop faster as the price increases due to the conditions of the loan. The following two properties established by Demange and Gale [14] within their model are essential for the group strategyproof mechanism of [17]: (1) the existence of one-sided optimal outcomes, which follows from the lattice property; (2) the property that the man-optimal mechanism is group strategyproof for the men. Note that these properties also hold in the stable marriage model: property (1) holds for strict preferences [38, attributed to Conway], but fails with weak preferences [51, Chapter 2]; property (2) holds for strict preferences [20], as mentioned earlier.

In this paragraph, we provide a brief summary of the group strategyproof mechanism of [17]. We model the stable marriage market with indifferences as a special form of the model of Demange and Gale, which we call a "tiered-slope market". We set the utility functions of the women in a form similar to a buyer in the assignment game of Shapley and Shubik [55]: the utility that a woman j assigns to being matched to a man i and paying an amount p is equal to a constant multiplied by the valuation that j assigns to i as described in Section 1.2.1, plus a term that depends on the priority of i to break ties, minus p. We set the slopes of the utility functions of the men as powers of a large fixed number, hence the name *tiered-slope market*: if a man i highly prefers a woman j, he assigns a large exponent $a_{i,j}$ in the slope associated with the utility function with woman j, and thus expects a small amount of compensation. The reserve utilities of the agents are set accordingly to prevent any agent from being matched to an unacceptable partner in an individually rational matching. We first establish that Pareto-stability in the stable marriage market with indifferences follows from stability in the associated tiered-slope market. We then show that the utility achieved by any man in a man-optimal solution to the associated tiered-slope market uniquely determines the tier of preference to which that man is matched in the stable marriage market with indifferences. Using this result, and group strategyproofness of man-optimal mechanisms in the model of Demange and Gale, we are able to show that group strategyproofness for the men in the stable marriage market with indifferences is achieved by man-optimality in the associated tiered-slope market. Finally we show that a man-optimal outcome can be computed in polynomial time by using the algorithm of Dütting et al. [21]; this requires $O(n^5)$ arithmetic operations with $poly(n)$ precision, resulting in an algorithm slower than the $O(n^4)$-time algorithm of this dissertation.

Equivalence of the Two Mechanisms

In Section 6.3, we show that the set of outputs of the IUAP-based mechanism of this dissertation is equal to the set of outputs of the group strategyproof mechanism of [17]. Our approach is based on a technique used by Demange and Gale [14] to study various structural properties of their model, such as the lattice property. Demange and Gale analyze market instances in which the agents and their utility functions are fixed, while the reserve utilities vary. As noted by Roth and Sotomayor [51, Chapter 9], lowering the reserve utility of an agent is analogous to extending the preferences of an agent in the stable marriage model, a technique used to study structural properties of the stable marriage model. Building on this idea, for each iteration of our IUAP-based mechanism, we show that the greedy MWMs of the UAP maintained by our mechanism coincides with the man-optimal outcomes of the corresponding tiered-slope market of [17] where the reserve utilities are lowered just enough to “reveal” only the preferences that are present in the UAP.

Part I

Fast Algorithms for Special Cases of Unit-Demand Auctions

Chapter 2

Unit-Demand Auctions with Linear Edge Weights

This chapter provides our data structure and algorithm for the first problem mentioned in Section 1.1.1, namely the problem of computing a VCG outcome of a UDALEW. An abbreviated version of the results presented in this chapter appears in a conference publication [16].

In Section 2.1, we briefly review some related work. In Section 2.2, we formally define the problem we solve, and we introduce some useful definitions. In Section 2.3, we present an incremental framework for solving the problem. In Section 2.4, we present a basic algorithm within the framework of Section 2.3. In Section 2.5, building on the concepts introduced in Section 2.4, we give a high-level description of our fast algorithm. In Section 2.6, we introduce two data structures and we describe how to efficiently implement the algorithm of Section 2.5. In Section 2.7, we extend the incremental framework to compute the VCG prices, and we present an algorithm within that framework. Finally, in Section 2.8, we provide some concluding remarks.

2.1 Related Work

Given an undirected graph $G = (V, E)$, a *matching* of G is a subset M of E such that no two edges in M share an endpoint. If G is a weighted graph, we define the weight of a matching as the sum of the weights of its constituent edges. The problem of finding a maximum weight matching (MWM) of a weighted bipartite graph, also known as the "assignment problem" in operations research, is a basic and well-studied problem in combinatorial optimization. A classic algorithm for the assignment problem is the Hungarian method [39], which admits an $O(|V|^3)$-time implementation. For dense graphs with arbitrary edge weights, this time bound remains the fastest known. Fredman and Tarjan [28] introduce Fibonacci heaps, and by utilizing this data structure to speed up shortest path computations, they obtain a running time of $O(|V|^2 \log |V| + |E| \cdot |V|)$ for the maximum weight bipartite matching problem. When the edge weights are integers in $\{0, \ldots, N\}$, Duan and Su [19] give a scaling algorithm with running time $O(|E|\sqrt{|V|} \log N)$. In this chapter, we consider a restricted class of complete weighted bipartite graphs where the edge weights have a special structure. Section 1.1 gives pointers to results aimed at developing fast algorithms for matching problems in some other restricted classes of bipartite graphs.

The edge weights of the complete bipartite graphs that we study in this chapter (the graphs encoding the UDALEWs) can be represented using Monge matrices. An $n \times m$ matrix $C = (c_{ij})$ is called a Monge matrix if $c_{ij} + c_{rs} \leq c_{is} + c_{rj}$ for $1 \leq i < r \leq n$, $1 \leq j < s \leq m$. Burkard [7] provides a survey of the rich literature on applications of Monge structures in combinatorial optimization problems. When the edge weights of a bipartite graph can be represented using a Monge matrix, an optimal maximum cardinality matching can be found in $O(nm)$ time where n is the number of rows and m is the number of columns. If $n = m$ then the diagonal of the Monge matrix representing the edge weights gives a trivial solution. Aggarwal et al. [3] study several weighted bipartite matching problems where, aside being a Monge matrix, additional structural properties are assumed for the matrix representing the edge weights. The authors present an $O(n \log m)$-time divide and conquer algorithm for the case where the number of rows n is at most the number of columns m and each row is bitonic, i.e., each row is a non-increasing sequence followed by a non-decreasing sequence. If we represent the edge weights of the graphs encoding the UDALEWs using a matrix so that the rows correspond to the bids and the columns correspond to the items, then both the Monge property and the bitonicity property are satisfied; in fact each row is monotonic. However, we end up having more rows than columns, which renders the algorithm of [3] inapplicable for our problems. If we had more columns than rows, as assumed in [3], then we would have a trivial solution which could be constructed by sorting the bids with respect to their slopes (and intercepts to break ties). In summary, similar to [3], this chapter efficiently solves the weighted bipartite matching problem for Monge matrices having an additional structure on the rows. In contrast, the structural assumption we place on the rows is stronger than that of [3], and we require more rows than columns, whereas [3] requires the opposite.

2.2 Preliminaries

A *bid* is a triple $u = (\mathit{slope}, \mathit{intercept}, \mathit{id})$ where *slope* and *intercept* are real numbers, and *id* is an integer. We use the notation $u.\mathit{slope}$ and $u.\mathit{intercept}$ to refer to the first and second components of a bid u, respectively. The bids are ordered lexicographically.

An *item* is a pair $v = (\mathit{quality}, \mathit{id})$ where *quality* is a real number and *id* is an integer. We use the notation $v.\mathit{quality}$ to refer to the first component of an item v. The items are ordered lexicographically. For any bid u and any item v, we define $w(u, v)$ as $u.\mathit{intercept} + u.\mathit{slope} \cdot v.\mathit{quality}$.

For any set of bids U and any set of items V, we define the pair (U, V) as a *unit-demand auction with linear edge weights* (*UDALEW*). Such an auction represents a unit-demand auction instance where the set of bids is U, the set of items is V, and each bid u in U offers an amount $w(u, v)$ on each item v in V.

A UDALEW $A = (U, V)$ corresponds to a complete weighted bipartite graph G where left vertices are U, right vertices are V, and the weight of the edge between a left vertex u and a right vertex v is equal to $w(u, v)$. Hence, for a UDALEW, we use the standard graph theoretic terminology, alluding to the corresponding graph.

A *matching* of a UDALEW (U, V) is a set M of bid-item pairs where each bid (resp., item) in M belongs to U (resp., V) and no bid (resp., item) appears more than once in M. The *weight* of a matching M, denoted $w(M)$, is defined as the sum, over all bid-item pairs (u, v) in M, of $w(u, v)$.

In this chapter, we solve the problem of finding a VCG outcome (allocation and prices) for a given UDALEW A; a VCG allocation is any MWM of A, and we characterize the VCG prices in Section 2.7.2. We reduce the problem of finding an MWM to the problem of finding a maximum weight maximum cardinality matching (MWMCM) as follows: we enlarge the given UDALEW instance $A = (U, V)$ by adding $|V|$ dummy bids to U, each with intercept zero and slope zero; we compute an MWMCM M of the resulting UDALEW A'; we remove from M all bid-item pairs involving dummy bids.

We conclude this section with some definitions that prove to be useful in the remainder of Chapter 2. For any totally ordered set S — such as a set of bids, a set of items, or an ordered matching which we introduce below — we make the following definitions: any integer i is an *index* in S if $1 \leq i \leq |S|$; for any element e in S, we define the *index of* e *in* S, denoted $\mathit{index}(e, S)$, as the position of e in the ascending order of elements in S, where the index of the first (resp., last) element, also called the *leftmost* (resp., *rightmost*) element, is 1 (resp., $|S|$); $S[i]$ denotes the element with index i in S; for any two indices i and j in S such that $i \leq j$, $S[i:j]$ denotes the set $\{S[i], \ldots, S[j]\}$ of size $j-i+1$; for any two integers i and j such that $i > j$, $S[i:j]$ denotes the empty set; for any integer i, $S[\,:i]$ (resp., $S[i:\,]$) denotes $S[1:i]$ (resp., $S[i:|S|]$); a subset S' is a *contiguous* subset of S if $S' = S[i:j]$ for some $1 \leq i \leq j \leq |S|$.

For any matching M, we define $\mathit{bids}(M)$ (resp., $\mathit{items}(M)$) as the set of bids (resp., items) that participate in M. A matching M is *ordered* if M is equal to $\bigcup_{1 \leq i \leq |M|} \{(U[i], V[i])\}$ where U denotes $\mathit{bids}(M)$ and V denotes $\mathit{items}(M)$. The order of the pairs in an ordered matching is determined by the order of the bids (equivalently, items) of those pairs.

2.3 Incremental Framework

In this section, we present an incremental framework for the problem of finding an MWMCM of a given UDALEW $A = (U, V)$. As discussed below, it is a straightforward problem if $|U| \leq |V|$. Thus, the primary focus is on the case where $|U| > |V|$. We start with a useful definition and a simple lemma.

For any set of bids U and any set of items V such that $|U| = |V|$, we define $\mathit{matching}(U, V)$ as the ordered matching $\{(U[1], V[1]), \ldots, (U[|U|], V[|U|])\}$.

Lemma 2.3.1 below shows how to compute an MWMCM of a UDALEW where the number of bids is equal to the number of items. The proof follows from the rearrangement inequality [34, Section 10.2, Theorem 368].

Lemma 2.3.1. For any UDALEW $A = (U, V)$ such that $|U| = |V|$, the ordered matching $\mathit{matching}(U, V)$ is an MWMCM of A.

Corollary 2.3.2. For any UDALEW $A = (U, V)$ such that $|U| \geq |V|$, there exists an ordered MWMCM of A.

If $|U| < |V|$ in a given UDALEW (U, V), then it is straightforward to reduce the problem to the case where $|U| = |V|$. Let U' (resp., U'') denote the set of the bids in U having negative (resp., nonnegative) slopes. Then we find an MWMCM M' of the UDALEW $(U', V[\,:|U'|])$ and an MWMCM M'' of the UDALEW $(U'', V[|V| - |U''| + 1 :\,])$, and we combine M' and M'' to obtain an MWMCM of (U, V).

It remains to consider the problem of finding an MWMCM of a UDALEW (U, V) where $|U| > |V|$. The following is a useful lemma.

Lemma 2.3.3. Let $A = (U, V)$ be a UDALEW such that $|U| \geq |V|$. Let u be a bid that does not belong to U. Let M be an MWMCM of A and let U' denote $\mathit{bids}(M)$. Then, any MWMCM of the UDALEW $(U' + u, V)$ is an MWMCM of the UDALEW $(U + u, V)$.

Proof. Let A' denote the UDALEW $(U + u, V)$. Assume that the claim is false, i.e., assume that there exists an MWMCM of $(U' + u, V)$ which is not an MWMCM of A'. Then, it is easy to see that for any MWMCM of A', the set of bids matched by this MWMCM is not included in $U' + u$. Let M' be an MWMCM of A' such that $|\mathit{bids}(M') \setminus U'|$ is minimized, and let u' be a bid such that u' is matched in M' and u' does not belong to $U' + u$. The symmetric difference of M and M', denoted $M \oplus M'$, corresponds to a collection of vertex-disjoint paths and cycles. Since u' is matched in M' and unmatched in M, we deduce that it is an endpoint of some path, call it P, in this collection. The edges of P alternate between M' and M. Let X denote the edges of P that belong to M, and let X' denote the edges of P that belong to M'. We consider two cases.

Case 1: P is of odd length. Since u' is an endpoint of P, we deduce that $|X'| = |X|+1$. It follows that $(M \setminus X) \cup X'$ is a matching of A with cardinality one higher than that of M, a contradiction since M is an MWMCM of A.

Case 2: P is of even length. Thus $|X'| = |X|$ and $(M' \setminus X') \cup X$ is an MCM of A'; in what follows, we refer to this MCM as M''. Let W denote the total weight of the edges in X and let W' denote the total weight of the edges in X'. We consider three subcases.

Case 2.1: $W < W'$. Thus $(M \setminus X) \cup X'$ is an MCM of A with weight higher than that of M, a contradiction since M is an MWMCM of A.

Case 2.2: $W > W'$. Thus M'' is an MCM of A' with weight higher than that of M', a contradiction since M' is an MWMCM of A'.

Case 2.3: $W = W'$. Thus M'' is an MWMCM of A' and $\mathit{bids}(M'') = \mathit{bids}(M') - u' + u''$, where u'' is the endpoint of P that is matched in M. It follows that $|\mathit{bids}(M'') \setminus U'| = |\mathit{bids}(M') \setminus U'| - 1$, contradicting the definition of M'. □

Lemma 2.3.3 shows that the problem of finding an MWMCM of a UDALEW (U, V) where $|U| = |V| + k$ reduces to k instances of the problem of finding an MWMCM of a UDALEW where the number of bids exceeds the number of items by one. Below we establish an efficient incremental framework for solving the MWMCM problem based on this reduction.

For any ordered matching M and any bid u that does not belong to $\mathit{bids}(M)$, we define $\mathit{insert}(M, u)$ as the ordered MWMCM M' of the UDALEW

$A = (\mathit{bids}(M) + u, \mathit{items}(M))$ such that the bid that is left unassigned by M', i.e., $(\mathit{bids}(M)+u) \setminus \mathit{bids}(M')$, is maximum, where the existence of M' is implied by Corollary 2.3.2.

We want to devise a data structure that maintains a dynamic ordered matching M. When the data structure is initialized, it is given an ordered matching M', and M is set to M'; we say that the data structure has initialization cost $T(n)$ if initialization takes at most $T(|M'|)$ steps. Subsequently, the following two operations are supported: the *bid insertion* operation takes as input a bid u not in $\mathit{bids}(M)$, and transforms the data structure so that M becomes $\mathit{insert}(M, u)$; the *dump* operation returns a list representation of M. We say that the data structure has bid insertion (resp., dump) cost $T(n)$ if bid insertion (resp., dump) takes at most $T(|M|)$ steps.

Lemma 2.3.4. Let $\mathcal{D}$ be an ordered matching data structure with initialization cost $f(n)$, bid insertion cost $g(n)$, and dump cost $h(n)$. Let A be a UDALEW (U, V) such that $|U| \geq |V|$. Then an MWMCM of A can be computed in $O(f(|V|) + (|U| - |V|) \cdot g(|V|) + h(|V|))$ time.

Proof. Let U' be a subset of U such that $|U'| = |V|$. Let $\langle u_1, \ldots, u_{|U|-|U'|}\rangle$ be a permutation of the bids in $U \setminus U'$. For any integer i such that $0 \leq i \leq |U| - |U'|$, let U_i denote $U' \cup \{u_1, \ldots, u_i\}$. Remark: $U_0 = U'$ and $U_{|U|-|U'|} = U$. We now show how to use $\mathcal{D}$ to find an ordered MWMCM of the UDALEW $A = (U_{|U|-|U'|}, V)$. We initialize $\mathcal{D}$ with $M_0 = \mathit{matching}(U_0, V)$, which by Lemma 2.3.1 is an ordered MWMCM of the UDALEW (U_0, V). Then we iteratively insert bids $u_1, \ldots, u_{|U|-|U'|}$. Let M_i denote the ordered matching associated with $\mathcal{D}$ after i iterations, $1 \leq i \leq |U| - |U'|$. By the definition of bid insertion, M_i is an ordered MWMCM of the UDALEW $(bids(M_{i-1}) + u_i, V)$, and thus, is an MWMCM of the UDALEW (U_i, V) by induction on i and Lemma 2.3.3. Thus, a dump on $\mathcal{D}$ after completing all iterations returns an ordered MWMCM of A. The whole process runs in the required time since we perform one initialization, $|U| - |U'|$ bid insertions, and one dump. $\square$

In Section 2.4, we give a simple linear-time bid insertion algorithm assuming an array representation of the ordered matching. Building on the concepts introduced in Section 2.4, Section 2.5 develops an ordered matching data structure with initialization cost $O(n \log^2 n)$, bid insertion cost $O(\sqrt{n} \log^2 n)$, and dump cost $O(n)$ (Theorem 2.6.7). The results of Section 2.5, together with Lemma 2.3.4, yield the $O(m\sqrt{n} \log^2 n)$ MWMCM time bound.

Looking from an auction perspective, as discussed in Section 2.2, our goal is to compute a VCG allocation and pricing given a UDALEW. In Section 2.7, we show how to extend the data structure of Section 2.5 to maintain the VCG prices as each bid is inserted. The asymptotic time complexity of the operations remain the same; the additional computation for maintaining the VCG prices takes $O(\sqrt{n})$ time at each bid insertion, where n denotes the size of the matching maintained by the data structure.

2.4 A Basic Bid Insertion Algorithm

In this section, we describe a linear-time implementation of $\mathit{insert}(M, u)$ given an array representation of the ordered matching. The algorithm described here is not only useful because it introduces the concepts that the fast algorithm we introduce in Section 2.5 is built on, but also the same approach is used in certain "block scan" computations of that fast algorithm. We first introduce two functions that, in a sense evident by their definitions, restrict $\mathit{insert}(M, u)$ into two halves, left and right, of M split by u.

For any ordered matching M and any bid u that does not belong to $\mathit{bids}(M)$, we define $\mathit{insert}_L(M, u)$ (resp., $\mathit{insert}_R(M, u)$) as the ordered MCM M' of the UDALEW $A = (\mathit{bids}(M)+u, \mathit{items}(M))$ of maximum weight subject to the condition that the bid that is left unassigned by M', i.e., $(\mathit{bids}(M) + u) \setminus \mathit{bids}(M')$, is less (resp., greater) than u, where the ties are broken by choosing the MCM that leaves the maximum such bid unassigned; if no such MCM exists, i.e., u is less (resp., greater) than every bid in $\mathit{bids}(M)$, then $\mathit{insert}_L(M, u)$ (resp., $\mathit{insert}_R(M, u)$) is defined as M.

The following lemma characterizes $\mathit{insert}(M, u)$ in terms of $\mathit{insert}_L(M, u)$ and $\mathit{insert}_R(M, u)$; the proof directly follows from the definitions of $\mathit{insert}(M, u)$, $\mathit{insert}_L(M, u)$, and $\mathit{insert}_R(M, u)$.

Lemma 2.4.1. Let M be a nonempty ordered matching and let u be a bid that does not belong to $\mathit{bids}(M)$. Let M_L denote $\mathit{insert}_L(M, u)$ and let M_R denote $\mathit{insert}_R(M, u)$. Let W denote the maximum of $w(M_L)$, $w(M)$, and

$w(M_R)$. Then,

$$insert(M,u) = \begin{cases} M_R & \text{if } w(M_R) = W \\ M & \text{if } w(M) = W > w(M_R) \\ M_L & \text{otherwise.} \end{cases}$$

We now introduce some definitions that are used in Lemma 2.4.2 below to characterize $insert_L(M,u)$ and $insert_R(M,u)$.

For any ordered matching M and any two indices i and j in M, we define M_i^j as $matching(U - U[i], V - V[j])$, where U denotes $bids(M)$ and V denotes $items(M)$.

Let M be a nonempty ordered matching, let U denote $bids(M)$, and let V denote $items(M)$. Then we define $\Delta_L(M)$ as $w(M_1^{|M|}) - w(M)$, i.e., $\sum_{i=2}^{|U|} w(U[i], V[i-1]) - \sum_{i=1}^{|U|} w(U[i], V[i])$, and we define $\Delta_R(M)$ as $w(M_{|M|}^1) - w(M)$, i.e., $\sum_{i=1}^{|U|-1} w(U[i], V[i+1]) - \sum_{i=1}^{|U|} w(U[i], V[i])$. It is straightforward to see that $\Delta_L(M[i:j])$ and $\Delta_R(M[i:j])$ can be computed for any $1 \le i \le j \le |M|$ by the recurrences

$$\Delta_L(M[k-1:j]) = \Delta_L(M[k:j]) + w(U[k], V[k-1]) - w(U[k-1], V[k-1]) \tag{L1}$$

and

$$\Delta_R(M[i:k+1]) = \Delta_R(M[i:k]) + w(U[k], V[k+1]) - w(U[k+1], V[k+1]) \tag{R1}$$

with base cases $\Delta_L(M[j]) = -w(U[j], V[j])$ and $\Delta_R(M[i]) = -w(U[i], V[i])$, respectively.

Let M be a nonempty ordered matching. Letting W denote $\max_{1 \le i \le |M|} w(M_i^{|M|})$, we define $\Delta_L^*(M)$ as $W - w(M)$, and we define $\mathit{loser}_L(M)$ as

$$\max\left\{ i \mid w(M_i^{|M|}) = W \right\}.$$

Symmetrically, letting W' denote $\max_{1 \le i \le |M|} w(M_i^1)$, we define $\Delta_R^*(M)$ as $W' - w(M)$, and we define $\mathit{loser}_R(M)$ as

$$\max\left\{ i \mid w(M_i^1) = W' \right\}.$$

By Lemma 2.3.1 and by the definitions of $\Delta_L(M)$ and $\Delta_R(M)$, it is straightforward to see that $(\Delta_L^*(M), \mathit{loser}_L(M)) = \max_{1 \le i \le |M|}(\Delta_L(M[i:]), i)$ and $(\Delta_R^*(M), \mathit{loser}_R(M)) = \max_{1 \le i \le |M|}(\Delta_R(M[:i]), i)$ (the pairs compare lexicographically). Hence, $\Delta_L^*(M[i:j])$, $\mathit{loser}_L(M[i:j])$, $\Delta_R^*(M[i:j])$, and $\mathit{loser}_R(M[i:j])$ can be computed for any $1 \le i \le j \le |M|$ by the recurrences

$$\begin{aligned}&\big(\Delta_L^*(M[k-1:j]), \mathit{loser}_L(M[k-1:j])\big) = \\ &\quad \max\{\big(\Delta_L^*(M[k:j]), \mathit{loser}_L(M[k:j]) + 1\big), \big(\Delta_L(M[k-1:j]), 1\big)\},\end{aligned} \tag{L2}$$

and

$$\begin{aligned}&\big(\Delta_R^*(M[i:k+1]), \mathit{loser}_R(M[i:k+1])\big) = \\ &\quad \max\{\big(\Delta_R^*(M[i:k]), \mathit{loser}_R(M[i:k])\big), \big(\Delta_R(M[i:k+1]), k+2-i\big)\}\end{aligned} \tag{R2}$$

with base cases $\Delta^*_L(M[j]) = -w(U[j], V[j])$, $\Delta^*_R(M[i]) = -w(U[i], V[i])$, and $\mathit{loser}_L(M[j]) = \mathit{loser}_R(M[i]) = 1$.

Lemma 2.4.2. Let M be a nonempty ordered matching, let U denote $\mathit{bids}(M)$, let V denote $\mathit{items}(M)$, let u be a bid that does not belong to U, let k denote $\mathit{index}(u, U + u)$, let M_L denote $\mathit{insert}_L(M, u)$, and let M_R denote $\mathit{insert}_R(M, u)$. If $k > 1$, then M_L is equal to $M_i^{k-1} + (u, V[k-1])$ and $w(M_L) = w(M) + \Delta^*_L(M[\,:k-1]) + w(u, V[k-1])$ where i denotes $\mathit{loser}_L(M[\,:k-1])$; otherwise, $M_L = M$. If $k \le |M|$, then M_R is equal to $M_j^k + (u, V[k])$ and $w(M_R) = w(M) + \Delta^*_R(M[k:\,]) + w(u, V[k])$ where j denotes $\mathit{loser}_R(M[k:\,]) + k - 1$; otherwise, $M_R = M$.

Proof. We address the claim regarding M_L; the claim regarding M_R is symmetric. There is nothing to prove if $k = 1$, so assume that $k > 1$. Since both M and M_L are ordered, and since each bid in M that is greater than u is in M_L, it is easy to see that $M[k:\,] = M_L[k:\,]$, and thus $w(M_L) - w(M) = w(M_L[\,:k-1]) - w(M[\,:k-1])$. Then, since M_L is ordered, u is matched to $V[k-1]$ in M_L, and thus M_L is equal to $M_i^{k-1} + (u, V[k-1])$ for some $i < k$. The observations in the preceding two paragraphs and the definitions of M_L, Δ^*_L, and loser_L imply that $w(M_L) - w(M) = w(u, V[k-1]) + \Delta^*_L(M[\,:k-1])$ and the index i in the preceding sentence is equal to $\mathit{loser}_L(M[\,:k-1])$. □

Lemmas 2.4.1 and 2.4.2, together with (L1), (R1), (L2), and (R2), directly suggest a linear-time computation of $\mathit{insert}(M, u)$, as shown in Algorithm 2.1. If $\mathit{insert}_L(M, u)$ (resp., $\mathit{insert}_R(M, u)$) is not equal to M, then the algorithm computes the difference $w(\mathit{insert}_L(M, u)) - w(M)$ (resp.,

Algorithm 2.1 A linear-time implementation of bid insertion. The difference of the weight of an MWMCM of the UDALEW $A = (\mathit{bids}(M) + u, \mathit{items}(M))$ and that of M is equal to δ, and the maximum bid in $\mathit{bids}(M) + u$ that is unmatched in some MWMCM of A is u^*.

Input: An ordered matching M and a bid u that does not belong to $\mathit{bids}(M)$.
Output: $\mathit{insert}(M, u)$.

```
1: Let $U$ denote $\mathit{bids}(M)$ and let $V$ denote $\mathit{items}(M)$
2: $C \leftarrow \{(0, u)\}$
3: $k \leftarrow \mathit{index}(u, U + u)$
4: if $k > 1$ then
5:     for $i = k - 1$ down to 1 do
6:         Compute $\Delta_L(M[i : k-1])$ via (L1)
7:         Compute $\Delta_L^*(M[i : k-1])$ and $\mathit{loser}_L(M[i : k-1])$ via (L2)
8:     end for
9:     $C \leftarrow C + (w(u, V[k-1]) + \Delta_L^*(M[\,: k-1]), U[i])$ where $i = \mathit{loser}_L(M[\,: k-1])$
10: end if
11: if $k \le |M|$ then
12:     for $i = k$ to $|M|$ do
13:         Compute $\Delta_R(M[k : i])$ via (R1)
14:         Compute $\Delta_R^*(M[k : i])$ and $\mathit{loser}_R(M[k : i])$ via (R2)
15:     end for
16:     $C \leftarrow C + (w(u, V[k]) + \Delta_R^*(M[k :\,]), U[j])$ where $j = \mathit{loser}_R(M[k :\,]) + k - 1$
17: end if
18: $(\delta, u^*) \leftarrow$ the lexicographically maximum pair in $C$
19: return $\mathit{matching}(U + u - u^*, V)$
```

$w(\mathit{insert}_R(M, u)) - w(M))$ and adds a pair at line 9 (resp., line 16) to a set C where the first component is this difference, and the second component is the bid in $\mathit{bids}(M) + u$ that is left unassigned by $\mathit{insert}_L(M, u)$ (resp., $\mathit{insert}_R(M, u)$). Then by Lemma 2.4.1, the algorithm correctly returns $\mathit{insert}(M, u)$ by choosing the maximum pair of C at line 18.

2.5 A Superblock-Based Bid Insertion Algorithm

In this section, we describe an ordered matching data structure based on the concept of a "superblock", and we show how to use this data structure to obtain a significantly faster bid insertion algorithm than that presented in Section 2.4. Before beginning our formal presentation in Section 2.5.1, we provide a high-level overview of the main ideas. A reader interested in only the formal presentation may proceed to Section 2.5.1 without loss of continuity.

Recall that an ordered matching data structure maintains a dynamic ordered matching M. Let n denote $|M|$. We maintain a partition of the bids of M into contiguous "groups" of size $\Theta(\ell)$, where ℓ is a parameter to be optimized later. The time complexity of Algorithm 2.1 is linear because the **for** loops starting at lines 5 and 12 process bid-item pairs in M sequentially. Our rough plan is to accelerate the computations associated with this pair of loops by proceeding group-by-group. We can process a group in constant time if we are given six "auxiliary values" that depend on the "submatching" M' of M associated with the bids in the group, namely: $\Delta_L(M')$, $\Delta_R(M')$, $\Delta^*_L(M')$, $\Delta^*_R(M')$, $loser_L(M')$, and $loser_R(M')$. The auxiliary values associated with a group can be computed in $\Theta(\ell)$ time. A natural approach is to precompute these auxiliary values when a group is created or modified, or when the set of matched items associated with the group is modified. Unfortunately, a single bid insertion can cause each bid in a contiguous interval of $\Theta(n)$ bids to have a new matched item. For example, if a bid insertion introduces a "low" bid u and deletes a "high" bid u', then each bid between u and u' gets a new matched item one position to the right of its old matched item. Since a constant fraction of the groups might need to have their auxiliary values recomputed as a result of a bid insertion, the overall time complexity remains linear.

The preceding discussion suggests that it might be useful to have an efficient way to obtain the new auxiliary values of a group of bids when the corresponding interval of matched items is shifted left or right by one position. To this end, we enhance the precomputation associated with a group of bids as follows: Instead of precomputing only the auxiliary values corresponding to the group's current matched interval of items, we precompute the auxiliary values associated with shifts of $0, \pm 1, \pm 2, \ldots, \pm\Theta(\ell)$ positions around the current matched interval. That way, unless a group of bids is modified (e.g., due to a bid being deleted or inserted) we do not need to redo the precomputation with the group until it has been shifted $\Omega(\ell)$ times. Since the enhanced precomputation computes $\Theta(\ell)$ sets of auxiliary values instead of one set, a naive implementation of the enhanced precomputation has $\Theta(\ell^2)$ time complexity, leading once again to linear worst-case time complexity for bid insertion. We obtain a faster bid insertion algorithm by showing how to perform the enhanced precomputation in $O(\ell \log^2 \ell)$ time.

Our $O(\ell \log^2 \ell)$-time algorithm for performing the enhanced precomputation forms the core of our fast bid insertion algorithm. Here we briefly mention the main techniques used to perform the enhanced precomputation efficiently; the reader is referred to Section 2.6.1 for further details. A divide-and-conquer approach is used to compute the auxiliary values associated with the functions $loser_L$ and $loser_R$ in $O(\ell \log \ell)$ time; the correctness of this approach is based on a monotonicity result (see Lemmas2.6.3 and 2.6.4). A convolution-based approach is used to compute the auxiliary values based on Δ_L and Δ_R in $O(\ell \log \ell)$ time (see Lemma 2.6.2). The auxiliary values based on $loser_L$ (resp., $loser_R$) are used within a divide-and-conquer framework to compute the auxiliary values based on Δ_L^* (resp., Δ_R^*); in the associated recurrence, the overhead term is dominated by the cost of evaluating the same kind of convolution as in the computation of the auxiliary values based on Δ_L and Δ_R. As a result, the overall time complexity for computing the auxiliary values based on Δ_L^* and Δ_R^* is $O(\ell \log^2 \ell)$.

Section 2.5.1 introduces the concept of a "block", which is used to represent a group of bids together with a contiguous interval of items that includes all of the items matched to the group. Section 2.6.1 presents a block data structure. When a block data structure is "initialized" with a group of bids and an interval of items, the enhanced precomputation discussed in the preceding paragraph is performed, and the associated auxiliary values are stored in tables. A handful of "fields" associated with the block are also initialized; these fields store basic information such as the number of bids or items in the block. After initialization, the block data structure is read-only: Whenever a block needs to be altered (e.g., because a bid needs to be inserted/deleted, because the block needs to be merged with an adjacent block), we destroy the block and create a new one. The operations supported by a block may be partitioned into three categories: "queries", "lookups" and "scans". Each query runs in constant time and returns the value of a specific field. Each lookup runs in constant time and uses a table lookup to retrieve one of the precomputed auxiliary values. Each of the two linear-time scan operations (one leftgoing, one rightgoing) performs a naive emulation of one of the **for** loops of Algorithm 2.1; in the context of a given bid insertion, such operations are only invoked on the block containing the insertion position of the new bid.

Section 2.5.1 defines the concept of a superblock, which is used to represent an ordered matching as a sequence of blocks. A superblock-based ordered matching data structure is introduced in Section 2.6.2, where each of the constituent blocks is represented using the block data structure alluded to in the preceding paragraph. In Section 2.6, we simplify the presentation by setting the parameter ℓ to $\Theta(\sqrt{n})$. For this choice of ℓ, we show that bid insertion can be performed using $O(1)$ block initializations, $O(\sqrt{n})$ block queries, $O(\sqrt{n})$ block lookups, at most two block scans, and $O(\sqrt{n})$ additional overhead, resulting in an overall time complexity of $O(\sqrt{n}\log^2 n)$. In terms of the parameters ℓ and n, the approach of Section 2.6 can be generalized to perform bid insertion using $O(\lceil n/\ell^2 \rceil)$ block initializations, $O(n/\ell)$ block queries, $O(n/\ell)$ block lookups, at most two block scans, and $O(n/\ell)$ additional overhead; it is easy to verify that setting ℓ to $\Theta(\sqrt{n})$ minimizes the overall time complexity.

2.5.1 Blocks and Superblocks

We define a *block* B as a UDALEW (U, V) where $|U| \leq |V|$. For any block $B = (U, V)$, we define *shifts*(B) as $|V| - |U| + 1$. For any block $B = (U, V)$ and any integer t such that $1 \leq t \leq$ *shifts*(B), we define *matching*(B, t) as *matching*$(U, V[t : t + |U| - 1])$.

Let M be a nonempty ordered matching, let U denote *bids*(M), and let V denote *items*(M). Let m be a positive integer, and let $\langle a_0, \ldots, a_m \rangle$, $\langle b_1, \ldots, b_m \rangle$, and $\langle c_1, \ldots, c_m \rangle$ be sequences of integers such that $a_0 = 0$, $a_m = |U|$, and $1 \leq b_i \leq a_{i-1} + 1 \leq a_i \leq c_i \leq |U|$ for $1 \leq i \leq m$. Let B_i denote the block $(U[a_{i-1} + 1 : a_i], V[b_i : c_i])$ for $1 \leq i \leq m$. Then the list of blocks $S = \langle B_1, \ldots, B_m \rangle$ is a *superblock*, and we make the following additional definitions: *matching*(S) denotes M; *size*(S) denotes $|M|$; *bids*(S) denotes U; *items*(S) denotes V; *shift*(S, i) and *shift*(S, B_i) both denote $b_i - a_{i-1}$ for $1 \leq i \leq m$; *sum*(S, i) denotes a_i for $0 \leq i \leq m$; the leftmost block B_1 and the rightmost block B_m are the *boundary blocks*, the remaining blocks $B_2, \ldots, B_{m-1}$ are the *interior blocks*. Remark: For any superblock S, *matching*$(S) = \bigcup_{1 \leq i \leq |S|}$ *matching*$(S[i], \mathit{shift}(S, i))$.

2.5.2 Algorithm 2.2

We obtain a significantly faster bid insertion algorithm than Algorithm 2.1 by accelerating the computations associated with the **for** loops starting at lines 5 and 12. Recall that the first loop computes $\Delta_L^*(M[\,: k - 1])$ and $\mathit{loser}_L(M[\,: k - 1])$, and the second one computes $\Delta_R^*(M[k :\,])$ and $\mathit{loser}_R(M[k :\,])$. These two loops process a trivial representation of M pair-by-pair using the recurrences (L1), (R1), (L2), and (R2). We start by generalizing these recurrences; these generalizations allow us to compute the aforementioned values more efficiently by looping over a superblock-based representation of the matching block-by-block, instead of pair-by-pair.

Let M denote $\mathit{matching}(U, V)$, and let i, j, and k be three indices in M such that $i \le j < k$. Then the following equation generalizes (L1), and it is straightforward to prove by repeated application of (L1).

$$\Delta_L(M[i:k]) = \Delta_L(M[j+1:k]) + w(U[j+1], V[j]) + \Delta_L(M[i:j]). \quad \text{(L1}'\text{)}$$

We also give a generalization of (L2), where the proof follows from the definitions of Δ_L^* and loser_L.

$$\begin{aligned}&\big(\Delta_L^*(M[i:k]), \mathit{loser}_L(M[i:k])\big) = \\ &\quad \max\left\{\begin{aligned}&\big(\Delta_L^*(M[j+1:k]), \mathit{loser}_L(M[j+1:k]) + j + 1 - i\big), \\ &\big(\Delta_L^*(M[i:j]) + w(U[j+1], V[j]) + \Delta_L(M[j+1:k]), \\ &\quad \mathit{loser}_L(M[i:j])\big)\end{aligned}\right\}\end{aligned} \quad \text{(L2}'\text{)}$$

Let M denote $\mathit{matching}(U, V)$, and let i, j, and k be three indices in M such that $i < j \le k$. Symmetrically, the following equations generalize (R1) and (R2).

$$\Delta_R(M[i:k]) = \Delta_R(M[i:j-1]) + w(U[j-1], V[j]) + \Delta_R(M[j:k]), \quad \text{(R1}'\text{)}$$

and

$$\big(\Delta^*_R(M[i:k]), \mathit{loser}_R(M[i:k])\big) = \max\left\{\begin{array}{l}\big(\Delta^*_R(M[i:j-1]), \mathit{loser}_R(M[i:j-1])\big), \\ \big(\Delta^*_R(M[j:k]) + w(U[j-1], V[j]) + \Delta_R(M[i:j-1]), \\ \quad \mathit{loser}_R(M[j:k]) + j - i\big)\end{array}\right\} \tag{R2$'$}$$

We use (L1′) and (L2′) (resp., (R1′) and (R2′)) within a loop that iterates over a superblock-based representation of the matching block-by-block. In each iteration of the loop, we are able to evaluate the right-hand side of (L1′) and (L2′) (resp., (R1′) and (R2′)) in constant time because the terms involving $M[j+1:k]$ (resp., $M[i:j-1]$) are carried over from the previous iteration, and the terms involving $M[i:j]$ (resp., $M[j:k]$) are already stored in precomputed tables associated with the blocks of the superblock.

The high-level algorithm is given in Algorithm 2.2. The input is a superblock S that represents an ordered matching, denoted M (i.e., $\mathit{matching}(S) = M$), and a bid u that does not belong to $\mathit{bids}(S)$. The output is a superblock representing $\mathit{insert}(M, u)$. The unique bid u^* that is unmatched in $\mathit{insert}(M, u)$ is identified using the block-based framework alluded to above. After identifying u^*, if $u^* \neq u$, the algorithm invokes a subroutine $\textsc{Swap}(S, u^*, u)$ which, given a superblock S, a bid u^* that belongs to $\mathit{bids}(S)$, and a bid u that does not belong to $\mathit{bids}(S)$, returns a superblock that represents $\mathit{matching}(\mathit{bids}(S)+u-u^*, \mathit{items}(S))$. We present our implementation of Swap and analyze its time complexity in Sections 2.6.3 and 2.6.4. The correctness

Algorithm 2.2 A high-level bid insertion algorithm using the superblock-based representation of an ordered matching.

```
Input: A superblock $S$ and a bid $u$ that does not belong to $bids(S)$.
Output: A superblock $S'$ such that $matching(S') = insert(matching(S), u)$.
 1: Let $M$ denote $matching(S)$, let $U$ denote $bids(S)$, and let $V$ denote $items(S)$
 2: Let $S[i]$ be $(U_i, V_i)$ for $1 \leq i \leq |S|$
 3: $\sigma(i) \leftarrow sum(S, i)$ for $0 \leq i \leq |S|$
 4: $C \leftarrow \{(0, u)\}$
 5: $\ell \leftarrow |\{(U', V') \mid (U', V') \in S$ and $U'[1] < u\}|$
 6: $k \leftarrow$ if $\ell < 1$ then $1$ else $index(u, U_\ell + u) + 1 + \sigma(\ell - 1)$
 7: if $k > 1$ then
 8:     for $i = k - 1$ down to $\sigma(\ell - 1) + 1$ do
 9:         Compute $\Delta_L(M[i : k-1])$ via (L1)
10:         Compute $\Delta^*_L(M[i : k-1])$ and $loser_L(M[i : k-1])$ via (L2)
11:     end for
12:     for $i = \ell - 1$ down to 1 do
13:         Compute $\Delta_L(M[\sigma(i-1) + 1 : k-1])$ via (L1$'$)
14:         Compute $\Delta^*_L(M[\sigma(i-1) + 1 : k-1])$ and $loser_L(M[\sigma(i-1) + 1 : k-1])$ via (L2$'$)
15:     end for
16:     $C \leftarrow C + (w(u, V[k-1]) + \Delta^*_L(M[\,: k-1]), U[i])$ where $i = loser_L(M[\,: k-1])$
17: end if
18: if $k \leq |M|$ then
19:     for $i = k$ to $\sigma(\ell)$ do
20:         Compute $\Delta_R(M[k : i])$ via (R1)
21:         Compute $\Delta^*_R(M[k : i])$ and $loser_R(M[k : i])$ via (R2)
22:     end for
23:     for $i = \ell + 1$ to $|S|$ do
24:         Compute $\Delta_R(M[k : \sigma(i)])$ via (R1$'$)
25:         Compute $\Delta^*_R(M[k : \sigma(i)])$ and $loser_R(M[k : \sigma(i)])$ via (R2$'$)
26:     end for
27:     $C \leftarrow C + (w(u, V[k]) + \Delta^*_R(M[k :\,]), U[j])$ where $j = loser_R(M[k :\,]) + k - 1$
28: end if
29: $(\delta, u^*) \leftarrow$ the lexicographically maximum pair in $C$
30: return if $u^* \neq u$ then SWAP$(S, u^*, u)$ else $S$
```

of Algorithm 2.2 is established in Lemma 2.5.1, where it is shown that Algorithm 2.2 emulates the behavior of Algorithm 2.1.

Lemma 2.5.1. Algorithm 2.2 is correct.

Proof. Assume that, given a superblock S, a bid u^* that belongs to $\mathit{bids}(S)$, and a bid u that does not belong to $\mathit{bids}(S)$, $\textsc{Swap}(S, u^*, u)$ correctly returns a superblock that represents $\mathit{matching}(\mathit{bids}(S) + u - u^*, \mathit{items}(S))$. Let M denote $\mathit{matching}(S)$, let U denote $\mathit{bids}(S)$, and let V denote $\mathit{items}(S)$, as in the algorithm. First, the algorithm performs a scan over the blocks to compute an integer ℓ at line 5 so that each bid in each block $S[i]$ for $1 \leq i < \ell$ (resp., $\ell < i \leq |S|$) is less (resp., greater) than the new bid u. Then it is easy to see that the integer k computed at line 6 is equal to $\mathit{index}(u, U+u)$, as in Algorithm 2.1. It remains to show that $\Delta_L^*(M[\,:k-1])$, $\mathit{loser}_L(M[\,:k-1])$, $\Delta_R^*(M[k:\,])$, and $\mathit{loser}_R(M[k:\,])$ are computed correctly so that the set C is populated with the same pairs as in Algorithm 2.1, and thus Lemma 2.4.1 implies that the bid u^* in $\mathit{bids}(M)+u$ that is left out by $\mathit{insert}(M, u)$ is correctly identified by choosing the maximum pair of C at line 29, as in Algorithm 2.1, and that the superblock returned at line 30 represents $\mathit{insert}(M, u)$.

If $k > 1$ (resp., $k \leq |M|$), the algorithm proceeds to emulate the **for** loop of Algorithm 2.1 that starts at line 5 (resp., line 12) to compute $\Delta_L^*(M[\,:k-1])$ and $\mathit{loser}_L(M[\,:k-1])$ (resp., $\Delta_R^*(M[k:\,])$ and $\mathit{loser}_R(M[k:\,])$). We first discuss the emulation of the loop of Algorithm 2.1 that starts at line 5; this emulation is performed by two **for** loops in Algorithm 2.2. The first loop in Algorithm 2.2, which starts at line 8, is identical to the loop of Algorithm 2.1, except that it stops when the boundary of the submatching represented by block $S[\ell]$ is reached. Thus, by repeated application of (L1) and (L2), upon completion of this first loop, we have computed $\Delta_L(M')$, $\Delta^*_L(M')$, and $\mathit{loser}_L(M')$ where M' denotes $M[\sigma(\ell-1)+1 : k-1]$. Then the second **for** loop, which starts at line 12, resumes where the first one left off; however, it utilizes the superblock-based representation of the ordered matching to loop block-by-block. During the iterations of the second loop, for $i = \ell - 1$ down to 1, (L1′) and (L2′) are invoked by setting i, j, and k in these equations to $\sigma(i-1)+1$, $\sigma(i)$, and $k-1$, respectively; thus the submatching $M[i : j]$ in these equations corresponds to the submatching that the block $S[i]$ represents in S, i.e., $\mathit{matching}(S[i], \mathit{shift}(S, i))$. During such an iteration i, for $i = \ell - 1$ down to 1, the terms involving $M[i : j]$ in (L1′) and (L2′), i.e., the terms that are equal to $\Delta_L(\mathit{matching}(S[i], \mathit{shift}(S, i)))$, $\Delta^*_L(\mathit{matching}(S[i], \mathit{shift}(S, i)))$, and $\mathit{loser}_L(\mathit{matching}(S[i], \mathit{shift}(S, i)))$, are fetched from the precomputed tables associated with the block $S[i]$. Note that all the terms in these equations involving $M[j+1 : k]$ are carried over from the previous iteration, except for the first iteration, where they are already computed by the first **for** loop. Thus, upon completion of these two loops, we have computed $\Delta^*_L(M[\,:k-1])$ and $\mathit{loser}_L(M[\,:k-1])$.

The emulation of the second loop of Algorithm 2.1 (starting at line 12) that computes $\Delta^*_R(M[k:\,])$ and $\mathit{loser}_R(M[k:\,])$ can be argued symmetrically, where (R1′) and (R2′) are invoked in the **for** loop at line 23 by setting i, j, and k in these equations to k, $\sigma(i-1)+1$, and $\sigma(i)$, respectively. Then the submatching $M[j : k]$ in these equations corresponds to the submatching that the block $S[i]$ represents in S, i.e., $\mathit{matching}(S[i], \mathit{shift}(S, i))$. $\square$

2.6 Fast Implementation of Algorithm 2.2

In this section, we complete the discussion of our fast bid insertion algorithm by describing two data structures, giving the implementation details, and analyzing the running time. First, in Section 2.6.1, we present a block data structure that precomputes the auxiliary tables mentioned in Section 2.5.2 in quasilinear time, thus allowing lines 13, 14, 24, and 25 of Algorithm 2.2 to be performed in constant time. Then, in Section 2.6.2, we introduce a superblock-based ordered matching data structure that stores the blocks using the block data structure, where the sizes of the blocks are optimized to balance the cost of Swap with that of the remaining operations in Algorithm 2.2. Finally, in Sections 2.6.3 and 2.6.4, we present our efficient implementation of Swap, which constructs only a constant number of blocks, and analyze its time complexity.

2.6.1 Block Data Structure

Let S be a superblock on which a bid insertion is performed, let B be a block in S, and let M_t denote $\mathit{matching}(B, t)$ for $1 \leq t \leq \mathit{shifts}(B)$. The algorithm may query $\Delta_L(M_t)$, $\Delta_R(M_t)$, $\Delta^*_L(M_t)$, $\Delta^*_R(M_t)$, $\mathit{loser}_L(M_t)$, and $\mathit{loser}_R(M_t)$ for $t = \mathit{shift}(S, B)$. If B is part of the superblocks for a series of bid insertions, then these queries may be performed for various t values.

For a fast implementation of Algorithm 2.2, instead of individually computing these quantities at query time, we efficiently precompute them during the construction of the block and store them in the following six lists. We define $\Delta_L(B)$ as the list of size $\mathit{shifts}(B)$ such that $\Delta_L(B)[t]$ is equal to $\Delta_L(M_t)$ for $1 \leq t \leq \mathit{shifts}(B)$. We define the lists $\Delta_R(B)$, $\Delta^*_L(B)$, $\Delta^*_R(B)$, $\mathit{loser}_L(B)$, and $\mathit{loser}_R(B)$ similarly. The representation of a block $B = (U, V)$ simply maintains each of the following explicitly as an array: U, V, $\Delta_L(B)$, $\Delta_R(B)$, $\Delta^*_L(B)$, $\Delta^*_R(B)$, $\mathit{loser}_L(B)$, and $\mathit{loser}_R(B)$. In what follows, we refer to that representation as the *block data structure for B*. The block data structure is an integral part of the superblock-based ordered matching data structure which we introduce in the following section.

Theorem 2.6.1. Let B be a block (U, V). Then the block data structure for B can be constructed in $O(|V|\,(\log \mathit{shifts}(B) + \log^2 |U|))$ time.

The proof of Theorem 2.6.1 follows directly from Lemmas 2.6.2, 2.6.5, and 2.6.6 below.

Lemma 2.6.2. Let B be a block (U, V). Then $\Delta_L(B)$ and $\Delta_R(B)$ can be computed in $O(|V| \log |U|)$ time.

Proof. We address the computation of $\Delta_L(B)$, the computation of $\Delta_R(B)$ is symmetric. Let β denote $\Delta_L(B)$. We define the following two real-valued functions on the set of integers. Let $x(n)$ be $V[n+1].\mathit{quality} - V[n+2].\mathit{quality}$, if $0 \leq n < |V|-1$; 0, otherwise. Let $h(n)$ be $U[|U|-n].\mathit{slope}$, if $0 \leq n < |U|-1$; 0, otherwise. Let $y(n)$ denote the discrete convolution $(x * h)(n) = \sum_m h(m) \cdot$

$x(n-m)$. Then, for $1 \le t \le \mathit{shifts}(B)$,

$$\begin{aligned}
\beta[t] &= \sum_{1<i\le|U|} w(U[i], V[i+t-2]) - \sum_{1\le i\le|U|} w(U[i], V[i+t-1]) \\
&= -w(U[1], V[t]) + \\
&\quad \sum_{1<i\le|U|} U[i].\mathit{slope} \cdot (V[i+t-2].\mathit{quality} - V[i+t-1].\mathit{quality}) \\
&= -w(U[1], V[t]) + \sum_{1<i\le|U|} h(|U|-i) \cdot x(i+t-3) \\
&= -w(U[1], V[t]) + \sum_{0\le m<|U|-1} h(m) \cdot x(|U|-m+t-3) \\
&= -w(U[1], V[t]) + y(t+|U|-3),
\end{aligned}$$

where the convolution $y(n) = (x*h)(n)$ can be computed in $O(|V| \log |U|)$ time by computing $\Theta(|U|)$-size segments of $y(n)$ using fast circular convolution, and concatenating the segments together [47]. □

The next two lemmas establish a monotonicity result that is used to prove Lemma 2.6.5.

Lemma 2.6.3. Let B be a block (U, V) and let α denote $\mathit{loser}_L(B)$. Then for any integer t such that $1 \le t < \mathit{shifts}(B)$, $\alpha[t] \ge \alpha[t+1]$.

Proof. For the sake of contradiction, suppose $\alpha[t] < \alpha[t+1]$ for some t such that $1 \le t < \mathit{shifts}(B)$. Let M denote $\mathit{matching}(B, t)$ and let M' denote $\mathit{matching}(B, t+1)$. Let i denote $\alpha[t] = \mathit{loser}_L(M)$ and let i' denote $\alpha[t+1] = \mathit{loser}_L(M')$. Since $i = \mathit{loser}_L(M) < i'$, we have $\Delta_L(M[i:]) > \Delta_L(M[i':])$, which, together with (L1$'$), implies $\Delta_L(M[i:i']) > \Delta_L(M[i'])$, and hence

$$\sum_{i<\ell\leq i'} w(U[\ell], V[\ell+t-2]) - \sum_{i\leq\ell<i'} w(U[\ell], V[\ell+t-1]) > 0. \tag{2.1}$$

Since $i' = \mathit{loser}_L(M')$, we have $\Delta_L(M'[i:]) \leq \Delta_L(M'[i':])$, which, together with (L1$'$), implies $\Delta_L(M'[i:i']) \leq \Delta_L(M'[i'])$, and hence

$$\sum_{i<\ell\leq i'} w(U[\ell], V[\ell+t-1]) - \sum_{i\leq\ell<i'} w(U[\ell], V[\ell+t]) \leq 0. \tag{2.2}$$

Subtracting (2.1) from (2.2), we get

$$\begin{aligned}
0 > & \sum_{i<\ell\leq i'} [w(U[\ell], V[\ell+t-1]) - w(U[\ell], V[\ell+t-2])] \\
& - \sum_{i\leq\ell<i'} [w(U[\ell], V[\ell+t]) - w(U[\ell], V[\ell+t-1])] \\
= & \sum_{i\leq\ell<i'} U[\ell+1].\mathit{slope} \cdot (V[\ell+t].\mathit{quality} - V[\ell+t-1].\mathit{quality}) \\
& - \sum_{i\leq\ell<i'} U[\ell].\mathit{slope} \cdot (V[\ell+t].\mathit{quality} - V[\ell+t-1].\mathit{quality}) \\
= & \sum_{i\leq\ell<i'} (U[\ell+1].\mathit{slope} - U[\ell].\mathit{slope})(V[\ell+t].\mathit{quality} - V[\ell+t-1].\mathit{quality}),
\end{aligned}$$

which contradicts the way that the bids in U and the items in V are ordered.

□

Lemma 2.6.4. Let B be a block (U,V) and let α denote $\mathit{loser}_R(B)$. Then for any integer t such that $1 \leq t < \mathit{shifts}(B)$, $\alpha[t] \leq \alpha[t+1]$.

Proof. Symmetric to the proof of Lemma 2.6.3. □

We now introduce two definitions that return "subblocks" of a block and that are useful in the proofs of Lemmas 2.6.5 and 2.6.6 which give divide-and-conquer algorithms.

For any block $B = (U, V)$ and for any two indices i and i' such that $1 \leq i \leq i' \leq |U|$, we define $\mathit{subBids}(B, i, i')$ as the block B' such that $\mathit{shifts}(B') = \mathit{shifts}(B)$ and $\mathit{matching}(B', t) = M_t[i : i']$ for $1 \leq t \leq \mathit{shifts}(B)$, where M_t denotes $\mathit{matching}(B, t)$; it is straightforward to see that $\mathit{subBids}(B, i, i') = (U[i : i'], V[i : |V| - |U| + i'])$.

For any block $B = (U, V)$ and for any two integers t and t' such that $1 \leq t \leq t' \leq \mathit{shifts}(B)$, we define $\mathit{subShifts}(B, t, t')$ as the block B' such that $\mathit{shifts}(B') = t' - t + 1$ and $\mathit{matching}(B', t'') = \mathit{matching}(B, t'' + t - 1)$ for $1 \leq t'' \leq \mathit{shifts}(B')$; it is straightforward to see that $\mathit{subShifts}(B, t, t') = (U, V[t : |V| - \mathit{shifts}(B) + t'])$.

Lemma 2.6.5. Let B be a block (U, V). Then $\mathit{loser}_L(B)$ and $\mathit{loser}_R(B)$ can be computed in $O(|V| \log \mathit{shifts}(B))$ time.

Proof. We address the computation of $\mathit{loser}_L(B)$, which relies on Lemma 2.6.3. The computation of $\mathit{loser}_R(B)$ is symmetric, and relies on Lemma 2.6.4. We begin by stating a useful claim.

Let M be an ordered matching and let j be an index in M. Then we claim that, $\mathit{loser}_L(M) \leq j$ implies $\mathit{loser}_L(M) = \mathit{loser}_L(M[\,:j])$; similarly, $\mathit{loser}_L(M) \geq j$ implies $\mathit{loser}_L(M) = \mathit{loser}_L(M[j:\,]) + j - 1$. The proof of the claim is immediate from (L2′).

Let α denote $\mathit{loser}_L(B)$. We give a divide-and-conquer algorithm that computes α. If $|U| = 1$, then $\alpha[t] = 1$ for any t and we are done; otherwise, we proceed as follows. Let t^* denote $\lceil \mathit{shifts}(B)/2 \rceil$ and let m denote $\alpha[t^*]$. We first compute m in $O(|U|)$ time using (L1) and (L2). Let B_1 denote the block $\mathit{subBids}(B, 1, m)$ and let B_2 denote the block $\mathit{subBids}(B, m, |U|)$. Let α_1 denote $\mathit{loser}_L(B_1)$ and let α_2 denote $\mathit{loser}_L(B_2)$. Then, by Lemma 2.6.3 and by the claim of the preceding paragraph,

$$\alpha[t] = \begin{cases} \alpha_1[t] & \text{if } t^* < t \leq \mathit{shifts}(B) \\ m & \text{if } t = t^* \\ \alpha_2[t] + m - 1 & \text{if } 1 \leq t < t^*. \end{cases}$$

Thus, it remains to compute $\alpha_1[t^* + 1 : \mathit{shifts}(B)]$ and $\alpha_2[1 : t^* - 1]$. Note that $\alpha_1[t^* + 1 : \mathit{shifts}(B)]$ is equal to $\mathit{loser}_L(B_1')$ for the block $B_1' = \mathit{subShifts}(B_1, t^* + 1, \mathit{shifts}(B))$, so we compute it recursively. Similarly, $\alpha_2[1 : t^* - 1]$ is equal to $\mathit{loser}_L(B_2')$ for the block $B_2' = \mathit{subShifts}(B_2, 1, t^* - 1)$, so we compute it recursively.

The overall running time satisfies the recurrence

$$T(n, s) \leq T(m, s/2) + T(n - m + 1, s/2) + O(n + s),$$

where n denotes $|U|$ and s denotes $\mathit{shifts}(B)$ for the input block $B = (U, V)$. Solving this recurrence, we obtain the desired running time. □

Lemma 2.6.6. Let B be a block (U, V). Then $\Delta_L^*(B)$ can be computed in $O(|V| \log^2 |U|)$ time given $\mathit{loser}_L(B)$. Similarly, $\Delta^*_R(B)$ can be computed within the same time bound given $\mathit{loser}_R(B)$.

Proof. We address the computation of $\Delta^*_L(B)$, which relies on Lemma 2.6.3. The computation of $\Delta^*_R(B)$ is symmetric, and relies on Lemma 2.6.4.

Let α denote $\mathit{loser}_L(B)$ and let β^* denote $\Delta^*_L(B)$. We now give a divide-and-conquer algorithm that computes β^*. If $|U| \leq 2$, then β^* can be computed trivially in $O(\mathit{shifts}(B))$ time; otherwise, we proceed as follows. Let m denote $\lceil |U|/2 \rceil$. Let B_1 denote the block $\mathit{subBids}(B, 1, m)$ and let B_2 denote the block $\mathit{subBids}(B, m+1, |U|)$. Let α_1 denote $\mathit{loser}_L(B_1)$ and let β^*_1 denote $\Delta^*_L(B_1)$. Let α_2 denote $\mathit{loser}_L(B_2)$, let β^*_2 denote $\Delta^*_L(B_2)$, and let β_2 denote $\Delta_L(B_2)$. Lemma 2.6.3 and the claim in the beginning of the proof of Lemma 2.6.5 imply that there exists an integer t' such that $0 \leq t' \leq \mathit{shifts}(B)$, $\alpha[t] = \alpha_1[t]$ for $t' < t \leq \mathit{shifts}(B)$, and $\alpha[t] = \alpha_2[t] + m - 1$ for $1 \leq t \leq t'$; in what follows let t^* denote the largest such integer. Then, by (L2$'$),

$$\beta^*[t] = \begin{cases} \beta^*_1[t] + \beta_2[t] + w(U[m+1], V[m+t-1]) & \text{if } t^* < t \leq \mathit{shifts}(B) \\ \beta^*_2[t] & \text{if } 1 \leq t \leq t^*. \end{cases}$$

Thus, it remains to compute $\beta^*_1[t^*+1 : \mathit{shifts}(B)]$, $\beta^*_2[1 : t^*]$, and $\beta_2[t^*+1 : \mathit{shifts}(B)]$. Note that $\beta^*_1[t^*+1 : \mathit{shifts}(B)]$ is equal to $\Delta^*_L(B'_1)$ for the block $B'_1 = \mathit{subShifts}(B_1, t^*+1, \mathit{shifts}(B))$, so we compute it recursively. Similarly, $\beta^*_2[1 : t^*]$ is equal to $\Delta^*_L(B'_2)$ for the block $B'_2 = \mathit{subShifts}(B_2, 1, t^*)$, so we compute it recursively. Finally, $\beta_2[t^*+1 : \mathit{shifts}(B)]$ is equal to $\Delta_L(B''_2)$ for the block $B''_2 = \mathit{subShifts}(B_2, t^*+1, \mathit{shifts}(B))$, and we compute it in $O(|V| \log |U|)$

time by Lemma 2.6.2.

The overall running time satisfies the recurrence

$$T(n, s) \leq T(n/2, t) + T(n/2, s - t) + O((n + s) \log n),$$

where n denotes $|U|$ and s denotes $\mathit{shifts}(B)$ for the input block $B = (U, V)$. Solving this recurrence, we obtain the desired running time. □

2.6.2 Superblock-Based Ordered Matching

In this section, we introduce a data structure called a *superblock-based ordered matching* (*SOM*). A SOM represents an ordered matching M by maintaining a superblock S such that $\mathit{matching}(S) = M$, where S is stored as a list of block data structures as described in Section 2.6.1.

Theorem 2.6.7. The SOM has initialization cost $O(n \log^2 n)$, bid insertion cost $O(\sqrt{n} \log^2 n)$, and dump cost $O(n)$.

Theorem 2.6.7 states our main result, and is proved in Section 2.6.4. We first briefly mention key performance-related properties of the SOM that are used for our efficient implementation of Algorithm 2.2. Throughout the rest of this paragraph, let n denote the size of the matching represented by the SOM. We group the operations performed during Algorithm 2.2 into three categories: $\Delta_L(M'')$, $\Delta_R(M'')$, $\Delta^*_L(M'')$, $\Delta^*_R(M'')$, $\mathit{loser}_L(M'')$, and $\mathit{loser}_R(M'')$ queries for submatchings M'' of M; the remaining operations performed in lines 1 through 29; the SWAP operation. It is easy to see that Algorithm 2.2

does not modify the superblock, except during SWAP at line 30. When SWAP modifies the superblock, existing blocks are not modified; rather, some existing blocks are deleted, and some newly constructed blocks are inserted. Since the SOM stores the superblock as a list of block data structures as described in Section 2.6.1, all the values in the auxiliary tables $\Delta_L(B)$, $\Delta_R(B)$, $\Delta^*_L(B)$, $\Delta^*_R(B)$, $\mathit{loser}_L(B)$, and $\mathit{loser}_R(B)$ are available for each block B. Thus, the queries for $\Delta_L(M'')$, $\Delta_R(M'')$, $\Delta^*_L(M'')$, $\Delta^*_R(M'')$, $\mathit{loser}_L(M'')$, and $\mathit{loser}_R(M'')$ for each $1 \leq i \leq |S|$ can be answered in constant time, where M'' denotes $\mathit{matching}(S[i], \mathit{shift}(S,i))$. It is easy to see by inspecting the code of Algorithm 2.2 that the number of such queries is proportional to the number of blocks in the superblock. Furthermore, the running time of all the remaining operations performed in lines 1 through 29 is proportional to the maximum of (1) the number of blocks in the superblock, and (2) the maximum number of bids in any single block. We define the blocks in a SOM so that each block has $\Theta(\sqrt{n})$ bids and $\Theta(\sqrt{n})$ items, yielding a $\Theta(\sqrt{n})$-time implementation of lines 1 through 29, and so that SWAP can be implemented by constructing at most a constant number of blocks. Later in this section, we formally define the SOM, and we introduce two invariants that are related to these requirements. Then in Sections 2.6.3 and 2.6.4, we present a detailed $O(\sqrt{n}\log^2 n)$-time implementation of SWAP for the SOM that constructs at most a constant number of blocks. We begin with some useful definitions.

For any non-empty ordered matching M, we define $\mathit{slice}(M)$ as $\lceil\sqrt{n}\rceil$ where n denotes $|M|$, and we find it convenient to overload *slice* so that $\mathit{slice}(S)$

denotes $\mathit{slice}(\mathit{matching}(S))$.

Let S be a superblock and let B be a block that belongs to S. Then we define $\mathit{time}(S,B)$ as $\min(\mathit{shift}(S,B), \mathit{shifts}(B) - \mathit{shift}(S,B) + 1)$. Observe that $\mathit{matching}(B,t)$ is well-defined for all t such that $\mathit{shift}(S,B) - \mathit{time}(S,B) < t < \mathit{shift}(S,B) + \mathit{time}(S,B)$.

We now introduce two invariants that we maintain regarding the performance-related concerns mentioned above. The first invariant ensures that the number of bids in each block is at least $\sqrt{n}$ and less than $2\left\lceil\sqrt{n}\right\rceil$, thus there are at most $\sqrt{n}$ blocks, where n denotes $\mathit{size}(S)$. The second invariant ensures that there are not too many blocks B in the superblock S whose time $\mathit{time}(S,B)$ is low, thus Swap does not require more than a constant number of block constructions. We now formally define these two invariants.

For any superblock S, we define the predicate $P(S)$ to hold if for each block (U,V) in S, $\mathit{slice}(S) \leq |U| < 2 \cdot \mathit{slice}(S)$.

For any superblock S, we define the predicate $Q(S)$ to hold if for any ℓ such that $1 \leq \ell \leq \mathit{slice}(S)$, there are at most ℓ interior blocks B of S such that $\mathit{time}(S,B) \leq \ell$.

We say that a superblock S is *nice* if $P(S) \wedge Q(S)$.

We say that an ordered matching data structure $\mathcal{D}$ is a *superblock-based ordered matching* (*SOM*) if it represents an ordered matching M by maintaining a nice superblock S such that $\mathit{matching}(S) = M$, and the superblock S is stored as a list of block data structures that are described earlier in Section 2.6.1.

2.6.3 Block-Level Operations

Having defined the SOM, it remains to show how to implement Swap efficiently on the SOM. We describe Swap by means of four kinds of block-level operations: *refresh*, *split*, *merge*, and *exchange*. As stated earlier, the primary goal of Swap is to update the matching, and *exchange* establishes that. The other three operations, *refresh*, *split*, and *merge*, do not alter the matching; the purpose of these operations is to maintain the two invariants defined in Section 2.6.2. The common goal of the *split* and *merge* operations is to keep the number of bids in each block of a superblock S within a constant factor of $\mathit{slice}(S)$, and to keep $|S|$ at most $\mathit{slice}(S)$. In addition, for any block B that is created by any of these four block-level operations on a superblock S, $\mathit{shifts}(B)$ is within a constant factor of $\mathit{slice}(S)$. We define here what each of these operations establishes and we outline how these operations are chained together in order to achieve a Swap implementation, we defer the analysis of the running times to Section 2.6.4. We start with some useful definitions.

All of the four block-level operations operate by replacing one or two existing blocks with one or two new blocks. We now outline how the new blocks are chosen by these operations. The choice of the bid set of a new block directly depends on the type of the operation: it is equal to the bid set of the block to be replaced by a *refresh*, or it is one of the two halves of the bid set of the block to be replaced by a *split*, or it is the union of the bid sets of the two blocks to be replaced by a *merge*, or one bid is removed and/or one bid is added to the bid set of a block to be replaced by a *exchange*. The exact details are given in the paragraphs below that introduce the individual operations. Given the bid set of the block that is to be created, the choice of the item set of the block depends only on the matching that the superblock resulting from the operation represents. Thus, any new block that the block-level operations create can be expressed as a function of the resulting matching M and the set U of the bids that are involved in the block. This function $\mathit{fresh}(M, U)$ is defined as follows.

Let M be an ordered matching and let U be a contiguous subset of $\mathit{bids}(M)$. Let i denote $\mathit{index}(U[1], \mathit{bids}(M))$ and let V denote $\mathit{items}(M)$. Let $U_{\prec}$ denote the set $\{u' \mid u' \in \mathit{bids}(M) \wedge u' < U[1]\}$ and let $U_{\succ}$ denote the set $\{u' \mid u' \in \mathit{bids}(M) \wedge u' > U[|U|]\}$ (note that $|U_{\prec}| = i-1$ and $|U_{\succ}| = |M|-|U|-|U_{\prec}|$). Then, we define $\mathit{fresh}(M, U)$ as the block $(U, V[i-r : i+|U|-1+r])$ where r denotes $2 \cdot \min(\mathit{slice}(M), |U_{\prec}|, |U_{\succ}|)$. For any superblock S and any block B in S that is equal to $\mathit{fresh}(\mathit{matching}(S), U)$ for some U, we make the following two observations: (1) $\mathit{time}(S, B) = \max_{S'} \mathit{time}(S', B)$ where the maximum is taken over all possible superblocks that B can be a part of; (2) $\mathit{time}(S, B) = 1 + \mathit{slice}(S)$ unless the degenerate condition $\min(|U_{\prec}|, |U_{\succ}|) < \mathit{slice}(S)$ holds, where $U_{\prec}$ and $U_{\succ}$ are defined as earlier in this paragraph. It will be explained later that, at the end of each bid insertion, the degenerate condition mentioned in the preceding observation only holds for the boundary blocks.

We are now ready to introduce the four block-level operations that the SOM performs to modify the superblock that it maintains. In order to define what these operations establish, we introduce a function for each operation that takes a superblock as input (with additional arguments for *exchange*) and returns another one.

For any superblock S, we define $\mathit{refresh}(S)$ as the superblock that is identical to S except that, if it exists, the block $B = (U, V)$ among all interior blocks with the lowest $\mathit{time}(S, B)$, breaking ties by choosing the block with the lowest index, is replaced with $\mathit{fresh}(\mathit{matching}(S), U)$.

For any superblock S, we define $\mathit{split}(S)$ as the superblock that is identical to S except that, if it exists, the block $B = (U, V)$ with the lowest index among the ones satisfying $|U| \geq 2 \cdot \mathit{slice}(S)$ is replaced with two blocks $\mathit{fresh}(\mathit{matching}(S), U[\,:\,m])$ and $\mathit{fresh}(\mathit{matching}(S), U[m+1\,:\,])$, where m denotes $\lceil |U|/2 \rceil$.

For any superblock S, we define $\mathit{merge}(S)$ as the superblock that is identical to S except that, if it exists, the block $B = (U, V)$ with the lowest index among the ones satisfying $|U| < \mathit{slice}(S)$, and the block $B' = (U', V')$ with the lowest index among the at most two that are adjacent to B, are replaced with a single block $\mathit{fresh}(\mathit{matching}(S), U \cup U')$.

It is easy to see that $\mathit{matching}(\mathit{refresh}(S))$, $\mathit{matching}(\mathit{split}(S))$, and $\mathit{matching}(\mathit{merge}(S))$ are all equal to $\mathit{matching}(S)$. However, the following function returns a superblock that represents a matching that is different than the one its input represents, by exchanging an existing bid for a new bid.

Let S be a superblock such that $\mathit{time}(S, B) > 1$ for each interior block B in S, let u^* be a bid that belongs to $\mathit{bids}(S)$, let u be a bid that does not belong to $\mathit{bids}(S)$, and let M denote $\mathit{matching}(\mathit{bids}(S)-u^*+u, \mathit{items}(S))$. Then we define $\mathit{exchange}(S, u^*, u)$, which returns a superblock that represents M and that is identical to S with the exception of at most two blocks, as follows: let $B^* = (U^*, V^*)$ denote the block in S that contains u^*; let $B^\dagger = (U^\dagger, V^\dagger)$ denote the block with the lowest index among the ones in S satisfying that $U^\dagger+u$ is a contiguous subset of $\mathit{bids}(S)+u$; if $B^* = B^\dagger$, then $\mathit{exchange}(S, u^*, u)$ is identical to S except that B^* is replaced with $\mathit{fresh}(M, U^* - u^* + u)$, otherwise, $\mathit{exchange}(S, u^*, u)$ is identical to S except that B^* is replaced with $\mathit{fresh}(M, U^* - u^*)$ and $B^\dagger$ is replaced with $\mathit{fresh}(M, U^\dagger + u)$.

In order to justify that $\mathit{exchange}(S, u^*, u)$ returns a valid superblock that represents the desired matching, we now compare $\mathit{matching}(S)$ with the desired matching from the perspectives of bids and blocks. In what follows, let S be a superblock and let u^* and u be two bids such that $\mathit{exchange}(S, u^*, u)$ is well-defined, and let S' denote $\mathit{exchange}(S, u^*, u)$. Let U denote $\mathit{bids}(S)$, let V denote $\mathit{items}(S)$, let M denote $\mathit{matching}(S)$, and let M' denote the desired matching $\mathit{matching}(U-u^*+u, V)$. Let k^* denote $\mathit{index}(u^*, U)$ and let k denote $\mathit{index}(u, U+u)$. Let B^* and $B^\dagger$ be the blocks defined as in the preceding paragraph, let ℓ^* denote $\mathit{index}(B^*, S)$, and let $\ell^\dagger$ denote $\mathit{index}(B^\dagger, S)$. Comparing M with M' from the perspective of bids, it is straightforward to see that, if $k^* < k$ (resp., $k^* \geq k$) then u is assigned to $V[k-1]$ (resp., $V[k]$) in M', and for each i such that $k^* < i \leq k-1$ (resp., $k \leq i < k^*$), the bid $U[i]$, which is assigned to $V[i]$ in M, is *shifted left* (resp., *right*) *by* $\mathit{exchange}(S, u^*, u)$, i.e., is assigned to $V[i-1]$ (resp., $V[i+1]$) in M'. Each bid that belongs to U but that is neither shifted left nor right by $\mathit{exchange}(S, u^*, u)$ is assigned to the same item in both M and M', except for u^* which is unassigned in M'. Comparing M with M' from the perspective of the blocks, it is easy to see that, if $\ell^* < \ell^\dagger$ (resp., $\ell^* > \ell^\dagger$), then each bid in each block $S[i]$ with a block index $\ell^* < i < \ell^\dagger$ (resp., $\ell^* > i > \ell^\dagger$) is shifted left (resp., right), hence, we say that the block $S[i]$ is *shifted left* (resp., *right*) *by* $\mathit{exchange}(S, u^*, u)$. For each block B that is shifted left (resp., right) by $\mathit{exchange}(S, u^*, u)$, $\mathit{shift}(S', B) = \mathit{shift}(S, B) - 1$ (resp., $+1$), and for each block B that belongs to both S and S' but that is neither shifted left nor right, $\mathit{shift}(S', B) = \mathit{shift}(S, B)$. Thus, since B^* and $B^\dagger$ are replaced with new blocks in S' and since $\mathit{time}(S, B) > 1$ for each interior block B that belongs to S, $\mathit{matching}(B, \mathit{shift}(S', B))$ is well-defined for each block B that belongs to S'. Hence, S' is a valid superblock and it is easy to see that $\mathit{matching}(S') = M'$.

We now define a function via *refresh*, *exchange*, *split*, and *merge* that proves to be useful in our goal of efficiently implementing Swap on the SOM.

For any nice superblock S, any bid u^* that belongs to $\mathit{bids}(S)$, and any bid u that does not belong to $\mathit{bids}(S)$, we define $\mathit{swap}(S, u^*, u)$ as

$$\mathit{split}(\mathit{merge}(\mathit{split}(\mathit{exchange}(\mathit{refresh}(S), u^*, u)))).$$

The following lemma suggests using $S = \mathit{swap}(S, u^*, u)$ as an implementation of $\textsc{Swap}(S, u^*, u)$ since it modifies the superblock as desired while maintaining the predicates $P(S)$ and $Q(S)$.

Lemma 2.6.8. Let S be a nice superblock, let u^* be a bid that belongs to

$\mathit{bids}(S)$, and let u be a bid that does not belong to $\mathit{bids}(S)$. Then $\mathit{swap}(S, u^*, u)$ is a nice superblock, and $\mathit{matching}(\mathit{swap}(S, u^*, u)) = \mathit{matching}(\mathit{bids}(S) + u - u^*, \mathit{items}(S))$.

Proof. Let S_1 denote $\mathit{refresh}(S)$. By the definition of *refresh*, $\mathit{matching}(S_1) = \mathit{matching}(S)$. Since S is nice, $P(S)$ and $Q(S)$ holds. Then, since *refresh* does not change the bid partitioning implied by the superblock, $P(S_1)$ holds. And since *refresh* only replaces an interior block B with the lowest $\mathit{time}(S, B)$, if it exists, with a block B' having $\mathit{time}(S_1, B') = \mathit{slice}(S_1)+1$, it is straightforward to see that a stronger $Q(S_1)$ (thus implying $Q(S_1)$) holds, which we define next as $Q^+(S_1)$.

For any superblock S, we define the predicate $Q^+(S)$ to hold if for any ℓ such that $1 \le \ell \le \mathit{slice}(S)$, there are at most $\ell - 1$ interior blocks B of S such that $\mathit{time}(S, B) \le \ell$.

Let S_2 denote $\mathit{exchange}(S_1, u^*, u)$. Since $Q^+(S_1)$ implies that $\mathit{time}(S_1, B)$ is greater than 1 for each interior block B in S_1, $\mathit{exchange}(S_1, u^*, u)$ is well-defined, and hence $\mathit{matching}(S_2) = \mathit{matching}(\mathit{bids}(S) + u - u^*, \mathit{items}(S))$. Now, if only one block is replaced during $\mathit{exchange}(S_1, u^*, u)$, then the following claims hold: $P(S_2)$ since the replaced block has the same number of bids; $Q^+(S_2)$, and thus $Q(S_2)$ since no blocks are shifted; hence, $\mathit{split}(\mathit{merge}(\mathit{split}(S_2)))$ is equal to S_2, which is a nice superblock with the desired matching, and we are done. If two blocks are replaced, then it is straightforward to see that the following claims hold by the definition of *exchange*: $Q(S_2)$ since $Q^+(S_1)$ and the fact that $|\mathit{shift}(S_2, B) - \mathit{shift}(S_1, B)| \le 1$ for each surviving block B, where the latter fact is a result of the shifts, as described while arguing the correctness of *exchange*; $P(S_2)$ except that one block may be undersized by one bid and one block may be oversized by one bid.

Let S_3 denote $\mathit{split}(S_2)$. It is straightforward to see that the following claims hold by the definition of *split*: $\mathit{matching}(S_3) = \mathit{matching}(S_2)$; $Q(S_3)$; $P(S_3)$ except that one block may be undersized by one bid.

Let S_4 denote $\mathit{merge}(S_3)$. It is straightforward to see that the following claims hold by the definition of *merge*: $\mathit{matching}(S_4) = \mathit{matching}(S_3)$; $Q(S_4)$; $P(S_4)$ except that one block may be oversized with total number of bids at most $3 \cdot \mathit{slice}(S) - 2$.

Let S_5 denote $\mathit{split}(S_4)$. It is straightforward to see that the following claims hold by the definition of *split*: $\mathit{matching}(S_5) = \mathit{matching}(S_4)$; $Q(S_5)$; $P(S_5)$.

By the preceding observations, we see that S_5, which is equal to $\mathit{swap}(S, u^*, u)$, is a nice superblock and $\mathit{matching}(S_5) = \mathit{matching}(S_2) = \mathit{matching}(\mathit{bids}(S) + u - u^*, \mathit{items}(S))$, as required. $\square$

2.6.4 Implementation of Swap and Time Complexity

We now complete the discussion of the fast bid insertion on the SOM by describing how to efficiently implement Swap as $S = \mathit{swap}(S, u^*, u)$, as described in the preceding section, and by proving Theorem 2.6.7, which summarizes our results. Recall that the goal of Swap(S, u^*, u) is, given a superblock S, a bid u^* that belongs to $\mathit{bids}(S)$, and a bid u that does not belong to $\mathit{bids}(S)$, to return a superblock that represents $\mathit{matching}(\mathit{bids}(S)+u-u^*, \mathit{items}(S))$; since a SOM always maintains a nice superblock, we require the input S and the returned superblock to be nice.

Lemma 2.6.9. Let $\mathcal{D}$ be a SOM, let S denote the superblock maintained by $\mathcal{D}$, and let n denote $\mathit{size}(S)$. Then, $\textsc{Swap}(S, u^*, u)$ on $\mathcal{D}$ can be implemented as $S = \mathit{swap}(S, u^*, u)$ in $O(\sqrt{n}\log^2 n)$ time.

Proof. Lemma 2.6.8 implies that $S = \mathit{swap}(S, u^*, u)$ is a correct implementation of $\textsc{Swap}(S, u^*, u)$, and it satisfies the requirement that $\mathcal{D}$ maintains a nice superblock. We now argue the running time. It is straightforward to see that each of the operations *refresh*, *exchange*, *split*, and *merge* can be implemented in $O(\sqrt{n}\log^2 n)$ time; it takes $O(|S|) = O(\sqrt{n})$ time to identify the block/blocks to be replaced, since $P(S)$ implies that $|S|$ is $\Theta(\sqrt{n})$; it takes $O(\sqrt{n}\log^2 n)$ time to construct each block $B = (U, V)$ by Theorem 2.6.1, since $P(S)$ and the definition of *fresh* implies that $|U|$, $|V|$, and $\mathit{shifts}(B)$ are $O(\sqrt{n})$; there are at most two block constructions per operation. $\square$

Proof of Theorem 2.6.7. When initialized with an ordered matching M with size n, the SOM constructs $\Theta(\sqrt{n})$ blocks, each taking $O(\sqrt{n}\log^2 n)$ time.

Bid insertion on the SOM can be implemented as Algorithm 2.2 in $O(\sqrt{n}\log^2 n)$ time since Lemma 2.6.9 shows that $\textsc{Swap}(S, u^*, u)$ can be implemented in $O(\sqrt{n}\log^2 n)$ time, and as argued in Section 2.6.2, the remaining operations in Algorithm 2.2 can be implemented in $O(\sqrt{n})$ time, where n denotes the size of the superblock that the SOM maintains.

It is straightforward to implement dump by scanning over all the blocks and constructing a list representation of the matching in $O(n)$ time where n denotes the size of the matching. $\square$

2.7 Computation of the VCG Prices

In this section, we show how to extend the SOM to maintain the VCG prices as each bid is inserted. Section 2.7.1 introduces some useful definitions. Section 2.7.2 extends the incremental framework of Section 2.3 to compute the VCG prices. Section 2.7.3 presents a basic algorithm within the framework of Section 2.7.2. Section 2.7.6 describes how to extend the data structure of Section 2.5 and presents a fast emulation of the algorithm of Section 2.7.3.

2.7.1 Preliminaries

We begin by reviewing some standard definitions and results that prove to be useful. We state these results for UDALEWs; however, they hold for general unit-demand auctions. (The reader is referred to [51, Chapter 8] for a thorough discussion and omitted proofs.)

For a UDALEW $A = (U, V)$, a *surplus vector* s assigns a real value $s[i]$ to each bid $U[i]$ in U, a *price vector* p assigns a real value $p[j]$ to each item $V[j]$ in V, and an *outcome* is a triple (M, s, p) such that s is a surplus vector, p is a price vector, and M is a matching of A.

An outcome (M, s, p) of a UDALEW (U, V) is *feasible* if $\sum_{1 \le i \le |U|} s[i] +$

$\sum_{1\le j\le |V|} p[j] = w(M)$. For any feasible outcome (M, s, p), we say that the pair of vectors (s, p) and the matching M are *compatible*.

Let $A = (U, V)$ be a UDALEW. We say that a bid $U[i]$ (resp., item $V[j]$) blocks an outcome (M, s, p) of A if $s[i] < 0$ (resp., $p[j] < 0$). We say that a bid-item pair $(U[i], V[j])$ blocks an outcome (M, s, p) of A if $s[i]+p[j] < w(U[i], V[j])$. If no bid, item, or bid-item pair blocks an outcome (M, s, p) of A , then we say that the outcome (M, s, p) is *stable*, and that the payoff (s, p) is *stable* with M. For any stable outcome (M, s, p) of A, the following are known: M is an MWM of A; $s[i] + p[j] = w(U[i], V[j])$ for all $(U[i], V[j])$ matched in M; $s[i] = 0$ for all $U[i]$ unmatched in M; $p[j] = 0$ for all $V[j]$ unmatched in M. It is also known that any MWM is compatible with any stable payoff. Thus, given the price vector p of a stable outcome of A, the corresponding surplus vector s is uniquely determined by the following equation, where M denotes an arbitrary MWM of A:

$$s[i] = \begin{cases} w(U[i], V[j]) - p[j] & \text{if } V[j] \text{ is assigned to } U[i] \text{ in } M \\ 0 & \text{if } U[i] \text{ is left unassigned in } M. \end{cases} \tag{2.3}$$

For any stable payoff (s, p) of A, we say that p is a *stable price vector* of A.

In the remainder of Section 2.7, we write an outcome as a pair (M, p) rather than a triple (M, s, p), and it is understood that the associated surplus vector s is given by (2.3).

It is known that the stable price vectors of a UDALEW form a lattice [55]. Hence, there is a unique stable price vector that is componentwise less than or equal to any other stable price vector; this minimum stable price vector corresponds to the VCG prices [40]. Thus, for a UDALEW A, we refer to a stable outcome (M, p) of A as a *VCG outcome of A* if p is the VCG prices. In the remainder of Section 2.7, the inequality operators denote componentwise inequalities when they are used on price vectors.

2.7.2 Incremental Framework with Prices

In this section, we present an incremental framework for the problem of finding a VCG outcome of a UDALEW; we follow the approach of Section 2.3. In order to utilize the algorithms of Sections 2.4 and 2.5, we assume that the UDALEW (U, V) for which we seek a VCG outcome is enlarged by adding $|V|$ dummy bids, each with intercept zero and slope zero, so that, by Corollary 2.3.2, we can restrict our attention to ordered MWMCMs. Hence, in the remainder of Section 2.7, for any outcome (M, p) of a UDALEW A, we impose the condition that M is an ordered MWMCM of A.

Let $A = (U, V)$ be a UDALEW such that $|U| \geq |V|$. Then for any VCG outcome (M, p) of A and any bid u that does not belong to U, we define $\mathit{insert}(M, p, u)$ as the stable outcome (M', p') of the UDALEW $A' = (\mathit{bids}(M) + u, \mathit{items}(M))$ where M' is $\mathit{insert}(M, u)$ and p' is the minimum stable price vector of A' such that $p' \geq p$; the existence and uniqueness of such p' is implied by the lattice property of the stable price vectors.

The following lemma is at the core of our incremental framework. The proof follows from [51, Proposition 8.17] and from Lemma 2.3.3.

Lemma 2.7.1. Let $A = (U, V)$ be a UDALEW such that $|U| \geq |V|$ and let u be a bid that does not belong to U. Then for any VCG outcome (M, p) of A, $\mathit{insert}(M, p, u)$ is a VCG outcome of the UDALEW $(U + u, V)$.

We want to devise a data structure that maintains a dynamic outcome (M, p). The data structure is initialized with a VCG outcome (M', p') of some UDALEW. The characterization of the data structure is analogous to that of Section 2.3, except that bid insertion transforms the data structure to represent $\mathit{insert}(M, p, u)$, and dump returns a list representation of both M and p.

Lemma 2.7.2. Let $\mathcal{D}$ be an outcome data structure with initialization cost $f(n)$, bid insertion cost $g(n)$, and dump cost $h(n)$. Let A be a UDALEW (U, V). Then a VCG outcome of A can be computed in $O(f(|V|) + (|U| - |V|) \cdot g(|V|) + h(|V|))$ time.

Proof. Let U' be a set of $|V|$ dummy bids, each with intercept zero and slope zero. Let $\langle u_1, \ldots, u_{|U|} \rangle$ be an arbitrary permutation of the bids in U. For any integer i such that $0 \leq i \leq |U|$, let U_i denote $U' \cup \{u_1, \ldots, u_i\}$. Remark: $U_0 = U'$ and $U_{|U|} = U \cup U'$. We now show how to use $\mathcal{D}$ to find a VCG outcome of the UDALEW $(U_{|U|}, V)$, which is also a VCG outcome of A. We initialize $\mathcal{D}$ with the outcome consisting of the ordered matching $M_0 = \mathit{matching}(U_0, V)$ and the all-zeros price vector p_0; note that (M_0, p_0) is a VCG outcome of the UDALEW (U_0, V). Then we iteratively insert bids $u_1, \ldots, u_{|U|}$. Let (M_i, p_i) denote the outcome associated with $\mathcal{D}$ after i iterations, $1 \leq i \leq |U|$. Then, by induction on i, Lemma 2.7.1 and the definition of bid insertion together imply that (M_i, p_i) is a VCG outcome of the UDALEW (U_i, V). Thus, a dump on $\mathcal{D}$ after completing all iterations returns a VCG outcome of A. The whole process runs in the required time since we perform one initialization, $|U|$ bid insertions, and one dump. $\square$

In Section 2.7.3, we give a linear-time bid insertion algorithm assuming an array representation of the ordered matching and the price vector. Building on the concepts introduced in Section 2.7.3 and the SOM of Section 2.5, Section 2.7.6 develops an outcome data structure with initialization cost $O(n \log^2 n)$, bid insertion cost $O(\sqrt{n} \log^2 n)$, and dump cost $O(n)$. The results of Section 2.7.6, together with Lemma 2.7.2, imply an $O(m\sqrt{n} \log^2 n)$ time bound for computing a VCG outcome.

2.7.3 A Basic Algorithm with Prices

In this section, we describe a linear-time implementation of *insert*(M, p, u) given an array representation of the ordered matching M and the price vector p. In Section 2.7.4, we give a characterization of the price component of *insert*(M, p, u); in Section 2.7.5, we show how to compute *insert*(M, p, u) based on this characterization. We start with some useful definitions and lemmas.

For any ordered matching M, we make the following definitions, where U denotes *bids*(M) and V denotes *items*(M): $U[1]$ (resp., $V[1]$) is the leftmost bid (resp., item) in M; $U[|M|]$ (resp., $V[|M|]$) is the rightmost bid (resp., item) in M; $V[j]$ is the *match of* $U[j]$ *in* M for $1 \leq j \leq |M|$; $V[j-1]$ is the *left-adjacent item of* $U[j]$ *in* M for $1 < j \leq |M|$; $V[j+1]$ is the *right-adjacent item of* $U[j]$ *in* M for $1 \leq j < |M|$; a bid-item pair consisting of a bid and its left-adjacent (resp., right-adjacent) item in M, i.e., $(U[j], V[j-1])$ (resp., $(U[j], V[j+1])$), is a *left-adjacent* (resp., *right-adjacent*) *pair in* M; a left-adjacent or a right-adjacent pair is also called an *adjacent pair*.

The following lemma plays a key role in our algorithm; it suggests that we focus on adjacent pairs to obtain a stable price vector.

Lemma 2.7.3. Let $A = (U, V)$ be a UDALEW such that $|U| \geq |V|$ and let M be an ordered MWMCM of A. Let p be a price vector such that no adjacent pair in M blocks the outcome (M, p) of A. Let u be a bid in U.

1. For any index i such that $1 \leq i < |M|$ and $u.slope \leq U[i].slope$, if $(u, V[i])$ does not block (M, p), then $(u, V[i+1])$ does not block (M, p).
2. For any index i such that $1 < i \leq |M|$ and $u.slope \geq U[i].slope$, if $(u, V[i])$ does not block (M, p), then $(u, V[i-1])$ does not block (M, p).

Proof. We prove the first claim; the second claim is symmetric. Let i be an index such that $1 \leq i < |M|$, $u.slope \leq U[i].slope$, and $(u, V[i])$ does not block (M, p). Since $V[i+1].quality \geq V[i].quality$ and $U[i].slope \geq u.slope$, we have

$$\begin{aligned} w(U[i], V[i+1]) - w(U[i], V[i]) &= U[i].slope \cdot (V[i+1].quality - V[i].quality) \\ &\geq u.slope \cdot (V[i+1].quality - V[i].quality) \\ &= w(u, V[i+1]) - w(u, V[i]). \end{aligned} \tag{2.4}$$

Since $(U[i], V[i{+}1])$ does not block (M, p), we have $p[i{+}1]-p[i] \geq w(U[i], V[i+1]) - w(U[i], V[i])$, and by (2.4), $w(u, V[i]) - p[i] \geq w(u, V[i+1]) - p[i+1]$. Since $(u, V[i])$ does not block (M, p), we know that the surplus of u is at least $w(u, V[i]) - p[i]$; combining this with the inequality established in the preceding sentence, we deduce that the surplus of u is at least $w(u, V[i+1]) - p[i+1]$, as required. □

For any outcome (M', p'), we make the following definitions, where U' denotes *bids*(M') and V denotes *items*(M'): a bid u in U' is *left-tight* (resp., *right-tight*) if it is indifferent between being assigned to its match in M' or being assigned to its left-adjacent (resp., right-adjacent) item in M'; for any two indices j_1 and j_2 such that $1 \leq j_1 < j_2 \leq |M'|$, the interval $[j_1, j_2]$ of (M', p') is *left-tight* if each bid $U'[j]$ for $j_1 < j \leq j_2$ is left-tight, and symmetrically, the interval $[j_1, j_2]$ of (M', p') is *right-tight* if each bid $U'[j]$ for $j_1 \leq j < j_2$ is right-tight.

For any outcome (M', p'), it is straightforward to observe the following, where U' denotes *bids*(M') and V denotes *items*(M'): if a bid $U'[j]$ is left-tight, then

$$p'[j-1] = p'[j] - w(U'[j], V[j]) + w(U'[j], V[j-1]), \tag{2.5}$$

and hence, if an interval $[j_1, j_2]$ is left-tight, then

$$p'[j_1] = p'[j_2] + \Delta_L(M'[j_1 : j_2]) + w(U'[j_1], V[j_1]); \tag{2.6}$$

symmetrically, if a bid $U'[j]$ is right-tight, then

$$p'[j+1] = p'[j] - w(U'[j], V[j]) + w(U'[j], V[j+1]), \tag{2.7}$$

and hence, if an interval $[j_1, j_2]$ is right-tight, then

$$p'[j_2] = p'[j_1] + \Delta_R(M'[j_1 : j_2]) + w(U'[j]_2, V[j_2]). \tag{2.8}$$

For any ordered matching M' and any real value t, we define $\mathit{tight}_L(M', t)$ (resp., $\mathit{tight}_R(M', t)$) as the price vector p' of the UDALEW (U, V) such that $p'[|V|] = t$ (resp., $p'[1] = t$), and for $j = |V|-1, \ldots, 1$ (resp., for $j = 2, \ldots, |V|$), $p'[j]$ is defined by (2.5) (resp., by (2.7)), where U' denotes $\mathit{bids}(M')$ and V denotes $\mathit{items}(M')$.

Let M' be an ordered matching, let V denote $\mathit{items}(M')$, let p be a price vector for V, and let u^* be a bid that does not belong to $\mathit{bids}(M')$. Then we define $\mathit{reach}_L(M', p, u^*)$ and $\mathit{reach}_R(M', p, u^*)$ as follows. Let j^* denote $\mathit{index}(u^*, \mathit{bids}(M') + u^*)$. If there exists an index j such that $1 \leq j < j^*$ and $p[j : j^* - 1] \leq \mathit{tight}_L(M'[j : j^* - 1], w(u^*, V[j^* - 1]))$, then $\mathit{reach}_L(M', p, u^*)$ is defined as the minimum such j; otherwise, $\mathit{reach}_L(M', p, u^*)$ is defined as j^*. Symmetrically, if there exists an index j such that $j^* \leq j \leq |M|$ and $p[j^* : j] \leq \mathit{tight}_R(M'[j^* : j], w(u^*, V[j^*]))$, then $\mathit{reach}_R(M', p, u^*)$ is defined as the maximum such j; otherwise, $\mathit{reach}_R(M', p, u^*)$ is defined as $j^* - 1$.

In the next section, we use the concepts introduced above to characterize the prices after bid insertion.

2.7.4 Characterization of the Prices After Bid Insertion

In this section, we fix an arbitrary outcome (M, p) that is stable for the UDALEW $(\mathit{bids}(M), \mathit{items}(M))$ and an arbitrary bid u that does not belong to $\mathit{bids}(M)$. In what follows, let U denote $\mathit{bids}(M)$, let V denote $\mathit{items}(M)$, let A denote the UDALEW $(U + u, V)$, let M' denote $\mathit{insert}(M, u)$, let U' denote $\mathit{bids}(M')$, let u^* denote $(U + u) \setminus U'$, let j^* denote $\mathit{index}(u^*, U + u)$, let $j_L^\dagger$ denote $\mathit{reach}_L(M', p, u^*)$, and let $j_R^\dagger$ denote $\mathit{reach}_R(M', p, u^*)$. We first introduce two useful lemmas and then, in Theorem 2.7.6, we characterize the prices after bid insertion.

Lemma 2.7.4. $M'[\,:j_L^\dagger - 1] = M[\,:j_L^\dagger - 1]$ and $M'[j_R^\dagger + 1:\,] = M[j_R^\dagger + 1:\,]$.

Proof. We prove by contradiction that $M'[\,:j_L^\dagger - 1] = M[\,:j_L^\dagger - 1]$; the proof of the other statement is symmetric. Assume that $M'[\,:j_L^\dagger - 1] \neq M[\,:j_L^\dagger - 1]$ and let k denote the least index such that $M'[k] \neq M[k]$; thus k is less than $j_L^\dagger$, which by definition is at most j^*. We consider two cases.

Case 1: $u \geq u^*$. Then $U'[\,:j^* - 1] = U[\,:j^* - 1]$, which implies $M'[\,:j^* - 1] = M[\,:j^* - 1]$, contradicting our assumption that $M'[\,:j_L^\dagger - 1] \neq M[\,:j_L^\dagger - 1]$.

Case 2: $u < u^*$. Then $u^* = U[j^* - 1]$, $u = U'[k]$, and $U'[k + 1 : j^* - 1] = U[k : j^* - 2]$ since M and M' are ordered matchings with $|M| - 1$ common bids and since $k < j^*$. The claim established below implies that $j_L^\dagger \leq k$, which contradicts $k < j_L^\dagger$, thereby completing the proof.

Claim: $p[k : j^* - 1] \leq \mathit{tight}_L(M'[k : j^* - 1], w(u^*, V[j^* - 1]))$. In what follows, let $q[j^* - 1]$ denote $w(u^*, V[j^* - 1])$ and let $q[j]$ denote $q[j + 1] -$

$w(U'[j+1], V[j+1]) + w(U'[j+1], V[j])$ for $k \leq j \leq j^* - 2$. In the remainder of the proof, we show by reverse induction on j that $p[j] \leq q[j]$ for $k \leq j < j^*$; then the claimed inequality follows immediately since the right-hand side is a vector with components $q[k], \ldots, q[j^* - 1]$.

Base case: $j = j^* - 1$. Since $u^* = U[j^* - 1]$ and since (M, p) is stable for the UDALEW $(\mathit{bids}(M), \mathit{items}(M))$, we have $p[j^* - 1] \leq w(u^*, V[j^* - 1]) = q[j^* - 1]$.

Induction step. Let j be an integer such that $k < j < j^*$ and assume $p[j+1] \leq q[j+1]$. Since the pair $(U[j], V[j+1])$ does not block (M, p), we know that $p[j] \leq p[j+1] - w(U[j], V[j+1]) + w(U[j], V[j])$. Then, by our assumption that $p[j+1] \leq q[j+1]$ and since $U'[j+1] = U[j]$, we deduce that $p[j] \leq q[j+1] - w(U'[j+1], V[j+1]) + w(U'[j+1], V[j]) = q[j]$. □

Lemma 2.7.5. For any item index j such that $j < j^*$, we have $j_L^\dagger \leq j$ if and only if $p[j] \leq w(u^*, V[j^* - 1]) + \Delta_L(M'[j : j^* - 1]) + w(U'[j], V[j])$. Symmetrically, for any item index j such that $j \geq j^*$, we have $j_R^\dagger \geq j$ if and only if $p[j] \leq w(u^*, V[j^*]) + \Delta_L(M'[j^* : j]) + w(U'[j], V[j])$.

Proof. We only prove the first claim; the proof of the second claim is symmetric. It is easy to see by the definition of $\mathit{reach}_L(M', p, u^*)$ that the claim holds for $j = j^* - 1$. We now show that if the claim holds for some item index j such that $1 < j < j^*$, then it holds for $j - 1$. In what follows, let $q[j]$ denote $w(u^*, V[j^* - 1]) + \Delta_L(M'[j : j^* - 1]) + w(U'[j], V[j])$ for $1 \leq j < j^*$. Let j be an item index such that $1 < j < j^*$ and assume that the claim holds for this index, i.e., $j_L^\dagger \leq j$ if and only if $p[j] \leq q[j]$. In what follows, let p' denote

$tight_L(M'[j-1 : j^*-1], w(u^*, V[j^*-1]))$. Note that $q[j] = p'[2]$ by (2.6), and $q[j-1] = q[j] - w(U'[j], V[j]) + w(U'[j], V[j-1]) = p'[1]$ by (L1$'$) and (2.6). We consider two cases.

Case 1: $j_L^\dagger \le j$ and $p[j] \le q[j]$. Since $j_L^\dagger \le j$, we know that $p[j : j^*-1] \le p'[2 :]$. Thus, $p[j-1] \le q[j-1]$ if and only if $p[j-1 : j^*-1] \le p'$. Hence, $j_L^\dagger \le j-1$ if and only if $p[j-1] \le q[j-1]$ by the definition of $reach_L(M', p, u^*)$.

Case 2: $j_L^\dagger > j$ and $p[j] > q[j]$. Then Lemma 2.7.4 implies that $U'[j] = U[j]$. The stability of (M, p) implies that $p[j-1] \ge p[j] - w(U[j], V[j]) + w(U[j], V[j-1])$. Then, since $p[j] > q[j]$ and $U'[j] = U[j]$, we conclude that $p[j-1] > q[j] - w(U'[j], V[j]) + w(U'[j], V[j-1]) = q[j-1]$. □

We now characterize a certain price vector given the stable outcome (M, p) and the new bid u, and then state in Theorem 2.7.6 that this price vector is the price component of any VCG outcome after insertion of u. We define $grow(M, p, u)$ as the price vector p' of A such that the following conditions hold: $p'[\ : j_L^\dagger - 1] = p[\ : j_L^\dagger - 1]$; if $j_L^\dagger < j^*$, then $p'[j_L^\dagger : j^*-1] = tight_L(M'[j_L^\dagger : j^*-1], w(u^*, V[j^*-1]))$; if $j_R^\dagger \ge j^*$, then $p'[j^* : j_R^\dagger] = tight_R(M'[j^* : j_R^\dagger], w(u^*, V[j^*]))$; $p'[j_R^\dagger + 1 :] = p[j_R^\dagger + 1 :]$.

Theorem 2.7.6. For any VCG outcome (M_0, p_0) of a UDALEW (U_0, V_0) such that $|U_0| \ge |V_0|$ and for any bid u_0 that does not belong to U_0, $insert(M_0, p_0, u_0) = (insert(M_0, u_0), grow(M_0, p_0, u_0))$.

Proof. The proof follows from Lemma 2.7.1, which summarizes our incremental framework, and from Lemma 2.7.7 below. □

The remainder of this section states and proves Lemma 2.7.7 which is used in the proof of Theorem 2.7.6. In what follows, let p' denote $\mathit{grow}(M, p, u)$.

Lemma 2.7.7. The following claims hold: (1) $p' \geq p$; (2) p' is a stable price vector of A; (3) for any stable price vector p'' of A such that $p'' \geq p$, $p' \leq p''$.

We state two useful lemmas before proving Lemma 2.7.7.

Lemma 2.7.8. $p'[j] \leq w(U'[j], V[j])$ for $1 \leq j \leq |V|$.

Proof. Since (M, p) is stable for the UDALEW (U, V), we know that $p[j] \leq w(U[j], V[j])$ for $1 \leq j \leq |V|$. Then, $p'[j] \leq w(U'[j], V[j])$ for $1 \leq j < j^\dagger_L$ (resp., $j^\dagger_R < j \leq |V|$) since $p'[\,:j^\dagger_L - 1] = p[\,:j^\dagger_L - 1]$ (resp., $p'[j^\dagger_R + 1 :\,] = p[j^\dagger_R + 1 :\,]$) by the definition of $\mathit{grow}(M, p, u)$, and since $U'[\,:j^\dagger_L - 1] = U[\,:j^\dagger_L - 1]$ (resp., $U'[j^\dagger_R + 1 :\,] = U[j^\dagger_R + 1 :\,]$) by Lemma 2.7.4. It remains to show that the claim holds for $j^\dagger_L \leq j \leq j^\dagger_R$. If $j^\dagger_L < j^*$ (resp., if $j^\dagger_R \geq j^*$), then $p'[j^* - 1] = w(u^*, V[j^* - 1]) \leq w(U'[j^* - 1], V[j^* - 1])$ (resp., $p'[j^*] = w(u^*, V[j^*]) \leq w(U'[j^*], V[j^*])$) where the equality holds by the definition of $\mathit{grow}(M, p, u)$ and the inequality holds by the fact that u^* is not matched by the MWMCM M'. Then it is straightforward to see that $p'[j] \leq w(U'[j], V[j])$ for $j = j^* - 2, \ldots, j^\dagger_L$ (resp., $j = j^* + 1, \ldots, j^\dagger_R$) since $p'[j]$ is defined by (2.5) (resp., by (2.7)). $\square$

Lemma 2.7.9. No adjacent pair in M' blocks the outcome (M', p') of A.

Proof. We start the proof by showing in the following three paragraphs that no adjacent pair in $M'[\,:j^* - 1]$ blocks the outcome (M', p'). The task of showing that no adjacent pair in $M'[j^* :]$ blocks (M', p') is symmetric. Then, we complete the proof by showing that if $1 < j^* \leq |M'|$, then neither of the two adjacent pairs in $M'[j^* - 1 : j^*]$ blocks the outcome (M', p').

First, we argue about the adjacent pairs in $M'[\,:\, j_L^\dagger - 1]$, which is nonempty only if $j_L^\dagger > 1$. Assume that $j_L^\dagger > 1$. Since $p'[\,: j_L^\dagger - 1] = p[\,: j_L^\dagger - 1]$ by the definition of $\mathit{grow}(M, p, u)$ and since $M'[\,:\, j_L^\dagger - 1] = M[\,:\, j_L^\dagger - 1]$ by Lemma 2.7.4, the stability of (M, p) implies that no pair (adjacent or not) in $M'[\,: j_L^\dagger - 1]$ blocks (M', p').

Second, we argue about the two adjacent pairs in $M'[j_L^\dagger - 1 : j_L^\dagger]$ when $j_L^\dagger < j^*$. Assume that $j_L^\dagger < j^*$. Since $p'[j_L^\dagger - 1] > p'[j_L^\dagger] - w(U'[j_L^\dagger], V[j_L^\dagger]) + w(U'[j_L^\dagger], V[j_L^\dagger - 1])$ by the definition of $j_L^\dagger$, the left-adjacent pair $(U'[j_L^\dagger], V[j_L^\dagger - 1])$ does not block (M', p'). Since $U'[j_L^\dagger - 1]$ is matched to the same item $(V[j_L^\dagger - 1])$ in M and in M' by Lemma 2.7.4, and since the right-adjacent pair $(U'[j_L^\dagger - 1], V[j_L^\dagger])$ does not block (M, p) (by the stability of (M, p)), we conclude that the same pair does not block (M', p') because the definition of $j_L^\dagger$ implies that $p'[j_L^\dagger - 1] = p[j_L^\dagger - 1]$ and $p'[j_L^\dagger] \geq p[j_L^\dagger]$.

Third, we argue about the adjacent pairs in $M'[j_L^\dagger : j^* - 1]$, which is nonempty only if $j_L^\dagger < j^* - 1$. Assume that $j_L^\dagger < j^* - 1$. Let j be an arbitrary index such that $j_L^\dagger < j < j^*$. It is easy to see that the left-adjacent pair $(U'[j], V[j-1])$ does not block (M', p') since $p'[j-1] = p'[j] - w(U'[j], V[j]) + w(U'[j], V[j-1])$ by (2.5). Since $U'[j-1] < U'[j]$, by an argument similar to the one that is used to derive (2.4), we deduce that $w(U'[j-1], V[j]) - w(U'[j-1], V[j-1]) \leq w(U'[j], V[j]) - w(U'[j], V[j-1])$, which combined with the equality in the preceding sentence implies that the right-adjacent pair $(U'[j-1], V[j])$ does not block (M', p').

Finally, we complete the proof by showing that if $1 < j^* \leq |M'|$, then neither of the two adjacent pairs in $M'[j^*-1 : j^*]$ blocks the outcome (M', p'). Assume that $1 < j^* \leq |M'|$. Then at least one of the following conditions hold: (1) $u \leq u^* < U'[j^*] = U[j^*]$; (2) $u \geq u^* > U'[j^*-1] = U[j^*-1]$. We argue about condition (1); the argument about condition (2) is symmetric. We start with some useful observations. Since $U'[j^*-1] \leq u^* < U[j^*]$, we deduce the following two inequalities by an argument similar to the one that is used to derive (2.4):

$$w(U'[j^*-1], V[j^*]) - w(U'[j^*-1], V[j^*-1]) \leq w(u^*, V[j^*]) - w(u^*, V[j^*-1]); \tag{2.9}$$

$$w(u^*, V[j^*]) - w(u^*, V[j^*-1]) \leq w(U[j^*], V[j^*]) - w(U[j^*], V[j^*-1]). \tag{2.10}$$

Stability of (M, p) implies the following two inequalities:

$$w(U[j^*-1], V[j^*]) - w(U[j^*-1], V[j^*-1]) \leq p[j^*] - p[j^*-1]; \tag{2.11}$$

$$p[j^*] - p[j^*-1] \leq w(U[j^*], V[j^*]) - w(U[j^*], V[j^*-1]). \tag{2.12}$$

It is easy to see that $U'[j^*-1] \leq U[j^*-1]$ because either $M = M'$ or

$U[j^* - 1] = u^*$; hence

$$\begin{aligned} w(U'[j^* - 1], V[j^*]) - w(U'[j^* - 1], V[j^* - 1]) \leq \\ w(U[j^* - 1], V[j^*]) - w(U[j^* - 1], V[j^* - 1]), \end{aligned} \tag{2.13}$$

by an argument similar to the one that is used to derive (2.9).

With these observations in mind, we want to show the stability of the right-adjacent pair in $M'[j^* - 1 : j^*]$, i.e.,

$$w(U'[j^* - 1], V[j^*]) - w(U'[j^* - 1], V[j^* - 1]) \leq p'[j^*] - p'[j^* - 1], \tag{2.14}$$

and the stability of the left-adjacent pair in $M'[j^* - 1 : j^*]$, i.e.,

$$p'[j^*] - p'[j^* - 1] \leq w(U[j^*], V[j^*]) - w(U[j^*], V[j^* - 1]), \tag{2.15}$$

where $p'[j^* - 1] = \max(p[j^* - 1], w(u^*, V[j^* - 1]))$ and $p'[j^*] = \max(p[j^*], w(u^*, V[j^*]))$. We consider three cases.

Case 1: $p'[j^* - 1] = w(u^*, V[j^* - 1])$ and $p'[j^*] = w(u^*, V[j^*])$. Then, (2.9) implies (2.14) and (2.10) implies (2.15).

Case 2: $p'[j^* - 1] = p[j^* - 1] \geq w(u^*, V[j^* - 1])$ and $p'[j^*] = w(u^*, V[j^*]) > p[j^*]$. Then $p[j^*] - p[j^* - 1] < p'[j^*] - p'[j^* - 1] \leq w(u^*, V[j^*]) - w(u^*, V[j^* - 1])$. The first inequality in the preceding sentence, (2.11), and (2.13) imply (2.14); the second inequality in the preceding sentence and (2.10) imply (2.15).

Case 3: $p'[j^* - 1] = w(u^*, V[j^* - 1]) > p[j^* - 1]$ and $p'[j^*] = p[j^*] \geq w(u^*, V[j^*])$. Then $w(u^*, V[j^*]) - w(u^*, V[j^* - 1]) \leq p'[j^*] - p'[j^* - 1] < p[j^*] -$

$p[j^*-1]$. The first inequality in the preceding sentence and (2.9) imply (2.14); the second inequality in the preceding sentence and (2.12) imply (2.15). □

Proof of Lemma 2.7.7. It is easy to see that claim (1) holds by the definition of $grow(M, p, u)$.

No item blocks the outcome (M', p') since the stability of p and claim (1) together imply that no price in p' is negative. No bid blocks the outcome (M', p') by Lemma 2.7.8. In order to prove claim (2), it remains to show that no bid-item pair in A blocks the outcome (M', p'). Observe that Lemmas 2.7.3 and 2.7.9 directly imply that no bid-item pair involving a bid in U' blocks (M', p'). Now, if $j^* > 1$ (resp., $j^* \leq |V|$), then it is easy to see that $(u^*, V[j^*-1])$ (resp., $(u^*, V[j^*])$) does not block (M', p') since $p'[j^*-1] \geq w(u^*, V[j^*-1])$ (resp., $p'[j^*] \geq w(u^*, V[j^*])$); thus, Lemmas 2.7.3 and 2.7.9 imply that no bid-item pair involving u^* blocks (M', p').

We now prove claim (3). Assume that there exists a stable price vector of A, denoted p'' in what follows, such that $p'' \geq p$ and $p''[j] < p'[j]$ for at least one item index j. We show a contradiction if $p''[j] < p'[j]$ for some $j \geq j^*$; the argument for the case where $j < j^*$ is symmetric. Assume that $p''[j] < p'[j]$ for some $j \geq j^*$ and let j' denote the minimum such j. Since $p'[j'] > p''[j'] \geq p[j']$, we conclude that $j_R^\dagger \geq j'$. We consider two cases.

Case 1: $j' = j^*$. Then $p''[j^*] < p'[j^*] = w(u^*, V[j^*])$, and thus the bid-item pair $(u^*, V[j^*])$ blocks the outcome (M', p'') since u^* is not matched by M', contradicting the stability of p''.

Case 2: $j' > j^*$. Then $p''[j'] < p'[j'] = p'[j'-1] - w(U'[j'-1], V[j'-$

$1]) + w(U'[j'-1], V[j'])$, where the equality holds by (2.7). Then, since the definition of j' implies $p'[j'-1] \leq p''[j'-1]$, we conclude that $p''[j'] < p''[j'-1] - w(U'[j'-1], V[j'-1]) + w(U'[j'-1], V[j'])$, and thus that the bid-item pair $(U'[j'-1], V[j])$ blocks the outcome (M', p''), contradicting the stability of p''. □

2.7.5 Computing Prices after Bid Insertion

In this section, we show how to compute $\mathit{insert}(M, p, u)$ in linear time. In what follows, let (M, p) be an arbitrary outcome that is stable for the UDALEW $(\mathit{bids}(M), \mathit{items}(M))$, let u be an arbitrary bid that does not belong to $\mathit{bids}(M)$, let M' denote $\mathit{insert}(M, u)$, let U denote $\mathit{bids}(M)$, let V denote $\mathit{items}(M)$, let U' denote $\mathit{bids}(M')$, let u^* denote $(U+u)\setminus U'$, let j^* denote $\mathit{index}(u^*, U+u)$, let $j_L^\dagger$ denote $\mathit{reach}_L(M', p, u^*)$, let $j_R^\dagger$ denote $\mathit{reach}_R(M', p, u^*)$, and let $p^\dagger$ denote $\mathit{grow}(M, p, u)$.

Algorithm 2.3 first computes M' and identifies the bid u^* that is not matched by M', using Algorithm 2.1 of Section 2.4. Then, Algorithm 2.3 computes $j_L^\dagger$ (resp., $j_R^\dagger$) in lines 6 through 15 (resp., 17 through 26), which we refer to as the *left-scan* (resp., *right-scan*) in what follows, by initializing the program variable j_L to j^* (resp., j_R to j^*-1), and then decrementing j_L (resp., incrementing j_R) until j_L is equal to $j_L^\dagger$ (resp., j_R is equal to $j_L^\dagger$). The prices $p^\dagger[j_L^\dagger : j_R^\dagger]$ are also computed on the fly during the left-scan (resp., right-scan) and stored in the array $p'[j_L^\dagger : j_R^\dagger]$. We begin our discussion by introducing two useful definitions.

Algorithm 2.3 A linear-time implementation of $\mathit{insert}(M, p, u)$.

```
Input: An ordered matching $M$ and a price vector $p$ such that $(M, p)$ is an
    outcome that is stable for the UDALEW $(\mathit{bids}(M), \mathit{items}(M))$, and a bid
    $u$ that does not belong to $\mathit{bids}(M)$.
Output: $\mathit{insert}(M, p, u)$.
 1: Let $U$ denote $\mathit{bids}(M)$ and let $V$ denote $\mathit{items}(M)$
 2: $M' \leftarrow \mathit{insert}(M, u)$
 3: $u^* \leftarrow (U + u) \setminus \mathit{bids}(M')$
 4: $j^* \leftarrow \mathit{index}(u^*, U + u)$
 5: $U' \leftarrow \mathit{bids}(M')$
 6: $j \leftarrow j_L \leftarrow j^*$
 7: while $j > 1$ do
 8:     $j \leftarrow j - 1$
 9:     $t \leftarrow$ if $j = j^* - 1$ then $w(u^*, V[j])$ else $p'[j+1] - w(U'[j+1], V[j+1]) + w(U'[j+1], V[j])$
10:     if $p[j] \leq t$ then
11:         $p'[j] \leftarrow t$
12:         $j_L \leftarrow j$
13:     else break
14:     end if
15: end while
16: $p'[\,: j_L - 1] = p[\,: j_L - 1]$
17: $j \leftarrow j_R \leftarrow j^* - 1$
18: while $j < |V|$ do
19:     $j \leftarrow j + 1$
20:     $t \leftarrow$ if $j = j^*$ then $w(u^*, V[j])$ else $p'[j-1] - w(U'[j-1], V[j-1]) + w(U'[j-1], V[j])$
21:     if $p[j] \leq t$ then
22:         $p'[j] \leftarrow t$
23:         $j_R \leftarrow j$
24:     else break
25:     end if
26: end while
27: $p'[j_R + 1 :\,] = p[j_R + 1 :\,]$
28: return $(M', p')$
```

We define the state predicate P_L to hold if $j_L^\dagger \le j_L$ and $p'[j_L : j^* - 1] = p^\dagger[j_L : j^* - 1]$. Symmetrically, we define the state predicate P_R to hold if $j_R^\dagger \ge j_R$ and $p'[j^* : j_R] = p^\dagger[j^* : j_R]$.

Lemma 2.7.10. The following claims hold: (1) the predicate P_L is an invariant of the **while** loop in the left-scan; (2) upon completion of the left-scan, $j_L = j_L^\dagger$.

Proof. The predicate P_L trivially holds upon execution of line 6, and thus at the beginning of the first iteration of the **while** loop, by the definition of $j_L^\dagger$. We prove by induction on the number of iterations that P_L holds at the end of each iteration. Consider an arbitrary iteration of the loop and assume that P_L holds at the beginning of the iteration. After executing line 8, let p'' denote $\mathit{tight}_L(M'[j : j^* - 1], w(u^*, V[j^* - 1]))$. If this is the first iteration of the loop, it is trivial to see that the variable t computed at line 9 is equal to $p''[1]$. Otherwise, since P_L holds at the beginning of the iteration, $p'[j + 1]$ is equal to $p''[2]$, and thus the variable t computed at line 9 is equal to $p''[1]$ by (2.5). Then it is easy to see that $j_L^\dagger \le j$ if and only if the condition $p[j] \le t$ at the **if** statement at line 10 holds, since $p[j + 1 : j^* - 1]$ is less than or equal to $p''[2 :]$ by our assumption that P_L holds at the beginning of the iteration. Then it is easy to see that P_L holds after execution of the **if** statement, and thus at the end of the iteration. It is also easy to see that $j_L = j_L^\dagger$ upon the termination of the loop: if the loop terminates via the **break** statement, then the inequality $j_L^\dagger > j = j_L - 1$ and P_L implies $j_L = j_L^\dagger$; if the loop terminates because $j = 1$ at line 7, then P_L implies $j_L = j_L^\dagger = j = 1$ since $j_L^\dagger \ge 1$ by definition. □

Lemma 2.7.11 below is symmetric to Lemma 2.7.10, and so its proof is omitted.

Lemma 2.7.11. The following claims hold: (1) the predicate P_R is an invariant of the **while** loop in the right-scan; (2) upon completion of the right-scan, $j_R = j_R^\dagger$.

Finally, the unchanged prices, i.e., $p^\dagger[\,:\, j_L^\dagger - 1]$ (resp., $p^\dagger[j_R^\dagger + 1 \,:\,]$), are copied into $p'[\,:\, j_L^\dagger - 1]$ (resp., $p'[j_R^\dagger + 1 \,:\,]$) after the left-scan (resp., right-scan) at line 16 (resp., line 27).

Lemma 2.7.12. Algorithm 2.3 is correct.

Proof. By Lemmas 2.7.10 and 2.7.11, and lines 16 and 27, the array p' is equal to $p^\dagger$ upon termination of the algorithm. The correctness follows by Theorem 2.7.6. □

2.7.6 Superblock-Based Price Computation

We obtain a fast emulation of Algorithm 2.3 by employing a superblock-based outcome representation and by extending the faster superblock-based bid insertion algorithm of Section 2.5.

For any nonempty ordered matching M, we say that π is a *price-block* for M if π is an array of real values of size $|M|$, or π is a pair (D, q) where $D \in \{L, R\}$ and q is a real value.

Let M be a nonempty ordered matching and let π be a price-block for M. Let U denote $\mathit{bids}(M)$ and let V denote $\mathit{items}(M)$. If π is an array, then we say that π explicitly represents prices, or π is an explicit price-block, and it is understood that $\pi[j]$ is the price of $V[j]$ for $1 \leq j \leq |V|$. If π is a pair (L, q) (resp., (R, q)), then we say that π compactly represents left-tight (resp., right-tight) prices, or π is a compact price-block, and it is understood that q is the price of $V[|V|]$ (resp., $V[1]$), and the prices of other items are defined by (2.6) (resp., (2.8)). If the price-block π compactly represents left-tight (resp., right-tight) prices, it is straightforward to see that the following claims hold: for any j such that $1 \leq j \leq |V|$, the price of $V[j]$ can be computed in $O(\min(j, |V| - j))$ time via (2.5) and (2.6) (resp., via (2.7) and (2.8)) given $\Delta_L(M)$ (resp., $\Delta_R(M)$); π can be converted to an explicit price-block in $O(|V|)$ time.

For any superblock S and any sequence $\Pi = \langle \pi_1, \ldots, \pi_{|S|} \rangle$ of price-blocks, we say that (S, Π) is a *superblock-based outcome* if π_i is a price-block for $\mathit{matching}(S[i], \mathit{shift}(S, i))$ for $1 \leq i \leq |S|$. It is understood that (S, Π) represents the outcome (M, p) where $M = \mathit{matching}(S)$ and the price vector p is represented by the individual price-blocks π_i. Thus, for any superblock-based outcome (S, Π) and any bid u that does not belong to $\mathit{bids}(S)$, we use the notation $\mathit{insert}(S, \Pi, u)$ to denote $\mathit{insert}(M, p, u)$, where (M, p) is the outcome that (S, Π) represents.

In order to obtain a fast implementation of $\mathit{insert}(M, p, u)$, we enhance the data structure of Section 2.6.2 so that we maintain a superblock-based outcome. We momentarily describe the necessary modifications in the implementation of the four block-level operations of Section 2.6.3, but we first characterize what our fast implementation computes.

Let (S, Π) be a superblock-based outcome and let u be a bid that does not belong to $\mathit{bids}(S)$. Let M' denote $\mathit{insert}(\mathit{matching}(S), u)$ and let u^* denote the bid $(\mathit{bids}(S) + u) \setminus M'$. Then for any superblock S' such that $\mathit{matching}(S') = M'$, $|S| = |S'|$, and $|\mathit{sum}(S, i) - \mathit{sum}(S', i)| \leq 1$ for $0 \leq i \leq |S|$, we define $\mathit{prices}(\Pi, S, u^*, S')$ as the set of all price-block sequences Π' such that the superblock-based outcome (S', Π') represents $\mathit{insert}(S, \Pi, u)$. Note that there may be more than one Π' in $\mathit{prices}(\Pi, S, u^*, S')$, each representing the same price vector, and we are only interested in computing an arbitrary one. We describe how to compute a price-block sequence that belongs to $\mathit{prices}(\Pi, S, u^*, S')$ in Section 2.7.8 by emulating Algorithm 2.3, starting with line 4.

2.7.7 Block-Level Operations

We are now ready to describe our efficient implementation of $\mathit{insert}(S, \Pi, u)$, which is based on an enhanced SOM. The input is a superblock-based outcome (S, Π) and a bid u that does not belong to $\mathit{bids}(S)$. We proceed as in Algorithm 2.2 of Section 2.5 and identify u^* at line 29. We enhance the four block-level operations, *refresh*, *split*, *merge*, and *exchange*, so that they operate on superblock-based outcomes, and thus Swap at line 30 modifies the superblock-based outcome (S, Π).

If the matching does not change, i.e., $u^* = u$ at line 30, we simply set the superblock-based outcome that the data structure maintains to (S', Π') where $S' = S$ and Π' is a price-block sequence that belongs to $\mathit{prices}(\Pi, S, u^*, S')$, and we are done. Otherwise, the enhanced block-level operations are performed during $\textsc{Swap}(S, u^*, u)$, as described next.

The *refresh* operation does not change the matching and does not change the partitioning of bids (by the blocks of S), so the price-block sequence Π does not need to be modified.

The *split* and *merge* operations do not change the matching, but they potentially change the partitioning of the bids, by either splitting a block or merging two blocks, respectively. If *split* replaces a block $S[i]$ with two blocks, then we perform the following two steps: if the corresponding price-block $\Pi[i]$ is compact, we convert it to explicit form; we split the array $\Pi[i]$ into two halves. If *merge* replaces two blocks $S[i]$ and $S[i+1]$ with a single block, then we perform the following two steps: if the corresponding price-block $\Pi[i]$ (resp., $\Pi[i+1]$) is compact, we convert $\Pi[i]$ (resp., $\Pi[i+1]$) to explicit form; we merge the arrays $\Pi[i]$ and $\Pi[i+1]$ into a single array.

The most complicated operation is *exchange* because it alters the matching. It is also at the end of the enhanced *exchange* where we emulate Algorithm 2.3 (starting with line 4) to compute the prices, as described in Section 2.7.8. The goal is to transform a superblock-based outcome (S, Π) into another superblock-based outcome (S', Π') such that $S' = \mathit{exchange}(S, u^*, u)$ (as described in Section 2.6.3) and Π' belongs to $\mathit{prices}(\Pi, S, u^*, S')$. First, the operation *exchange*(S, u^*, u) is carried out as described in Section 2.6.3 to obtain the superblock S', except that the replaced blocks are not discarded until the whole process we describe next is finished, since the blocks corresponding to the old superblock will be useful. Therefore, in what follows, we assume that both S and S' are available. Note that $|sum(S, i) - sum(S', i)| \leq 1$ for $0 \leq i \leq |S|$, as required by the definition of *prices*, where the difference of one is caused by the shifts and by the fact that one block may lose a bid and another block may gain a bid. Second, we compute a price-block sequence Π' that belongs to *prices*(Π, S, u^*, S'), and we set the superblock-based outcome that the data structure represents to (S', Π').

In the next section, we describe how to compute a price-block sequence that belongs to *prices*(Π, S, u^*, S') in $O(\sqrt{n})$ time, where n denotes *size*(S). It is straightforward to see that all the additional computations described above associated with enhanced *refresh*, *split*, *merge*, and *exchange* operations take $O(\sqrt{n})$ time.

2.7.8 Fast Update of Price-Blocks

In order to complete the description of our efficient implementation of *insert*(S, Π, u), we now show how to compute a price-block sequence that belongs to *prices*(Π, S, u^*, S') in $O(\sqrt{n})$ time. Our approach is to emulate Algorithm 2.3, starting with line 4, on input M, p, and u where (M, p) is the outcome that (S, Π) represents. In what follows, when we refer to the execution of Algorithm 2.3, we mean the execution associated with this input. In the remainder of this section, let M' denote *matching*(S'), which is equal to the ordered matching that the program variable of the same name stores during the execution of Algorithm 2.3, let U' denote *bids*(M'), let V denote *items*(M'), let $j_L^\dagger$ (resp., $j_R^\dagger$) denote $\mathit{reach}_L(M', p, u^*)$ (resp., $\mathit{reach}_R(M', p, u^*)$), which is equal to the value of j_L (resp., j_R) at the end of the execution of Algorithm 2.3, and let $p^\dagger$ denote $\mathit{grow}(M, p, u)$. Before beginning our formal presentation, we present the key idea underlying our fast emulation.

Recall that Algorithm 2.3 computes $j_L^\dagger$ (resp., $j_R^\dagger$) by initializing the program variable j_L to j^* (resp., j_R to $j^* - 1$), and then decrementing j_L (resp., incrementing j_R) by one at each iteration in a loop until j_L is equal to $j_L^\dagger$ (resp., j_R is equal to $j_L^\dagger$). Our fast emulation of Algorithm 2.3 relies on the following lemma, which allows us to "jump over" the blocks, i.e., decrement j_L by one plus the size of a block when we are at a block boundary.

Lemma 2.7.13. Let j_1 and j_2 be indices in M such that $j_1 < j_2$ and $j_L^\dagger \leq j_2$ (resp., $j_R^\dagger \geq j_1$). Then, we can decide whether $j_L^\dagger \leq j_1$ (resp., $j_R^\dagger \geq j_2$) holds and we can compute $p^\dagger[j_1]$ (resp., $p^\dagger[j_2]$) in constant time given $\Delta_L(M'[j_1 : j_2])$, $p[j_1]$, and $p^\dagger[j_2]$ (resp., $\Delta_R(M'[j_1 : j_2])$, $p[j_2]$, and $p^\dagger[j_1]$).

Proof. We address the claims regarding $j_L^\dagger$ and the computation of $p^\dagger[j_1]$; the claims regarding $j_R^\dagger$ and the computation of $p^\dagger[j_2]$ are symmetric. Let $\overline{p}$ denote $p^\dagger[j_2] + \Delta_L(M'[j_1 : j_2]) + w(U'[j_1], V[j_1])$, which can be computed in constant time given $p^\dagger[j_2]$ and $\Delta_L(M'[j_1 : j_2])$. Since $j_L^\dagger \leq j_2$, the definition of $p^\dagger$ implies that $p^\dagger[j_2] = w(u^*, V[j^* - 1]) + \Delta_L(M'[j_2 : j^* - 1]) + w(U'[j_2], V[j_2])$, which is equal to $w(u^*, V[j^*-1]) + \Delta_L(M'[j_2 + 1 : j^* - 1]) + w(U'[j_2+1], V[j_2])$ by (L1).

Then, $\overline{p}$ is equal to $w(u^*, V[j^* - 1]) + \Delta_L(M'[j_1 : j^* - 1]) + w(U'[j_1], V[j_1])$ by (L1$'$), and Lemma 2.7.5 implies that $j_L^\dagger \leq j_1$ if and only if $p[j_1] \leq \overline{p}$. Finally, the definition of $p^\dagger$ implies that $p^\dagger[j_1]$ is equal to $\overline{p}$ if $j_L^\dagger \leq j_1$, to $p[j_1]$ otherwise. □

Before presenting Algorithm 2.4, our fast implementation of computing a price-block sequence that belongs to $\textit{prices}(\Pi, S, u^*, S')$, we introduce several useful definitions.

For any superblock-based outcome (S, Π) and any index ℓ such that $1 \leq \ell \leq |S|$, we define $\textit{explicit}(S, \Pi, \ell)$ as the explicit price-block (an array of real values) that stores the same prices of $\textit{items}(\textit{matching}(S[\ell], \textit{shift}(S, \ell)))$ as in $\Pi[\ell]$. Note that $\textit{explicit}(S, \Pi, \ell)$ can be computed in $O(\textit{sum}(S, \ell) - \textit{sum}(S, \ell - 1))$ time.

For any superblock-based outcome (S, Π), any index ℓ such that $1 \leq \ell \leq |S|$, and any integer $\textit{offset}$ such that $\max(-1, 1-\ell) \leq \textit{offset} \leq 1$, we define $\textit{boundary}_L(S, \Pi, \ell, \textit{offset})$ as follows: if $\textit{offset} \geq 0$, then $\textit{boundary}_L(S, \Pi, \ell, \textit{offset})$ is the price of the item $V[1 + \textit{offset}]$ represented by $\Pi[\ell]$, where V denotes $\textit{items}(\textit{matching}(S[\ell], \textit{shift}(S, \ell)))$; if $\textit{offset} = -1$, then $\textit{boundary}_L(S, \Pi, \ell, \textit{offset})$ is the price of the item $V[|V|]$ represented by $\Pi[\ell - 1]$, where V denotes $\textit{items}(\textit{matching}(S[\ell - 1], \textit{shift}(S, \ell - 1)))$. Note that $\textit{boundary}_L(S, \Pi, \ell, \textit{offset})$ is the price of the item $V[\textit{sum}(S, \ell - 1) + 1 + \textit{offset}]$ represented by Π, where V denote $\textit{items}(S)$, and it can be computed in constant time.

Symmetrically, for any superblock-based outcome (S, Π), any index ℓ such that $1 \leq \ell \leq |S|$, and any integer $\textit{offset}$ such that $-1 \leq \textit{offset} \leq$

$\min(1, |S| - \ell)$, we define $\mathit{boundary}_R(S, \Pi, \ell, \mathit{offset})$ as follows: if $\mathit{offset} \leq 0$, then $\mathit{boundary}_R(S, \Pi, \ell, \mathit{offset})$ is the price of the item $V[|V| + \mathit{offset}]$ represented by $\Pi[\ell]$, where V denotes $\mathit{items}(\mathit{matching}(S[\ell], \mathit{shift}(S, \ell)))$; if $\mathit{offset} = 1$, then $\mathit{boundary}_R(S, \Pi, \ell, \mathit{offset})$ is the price of the item $V[1]$ represented by $\Pi[\ell+1]$, where V denotes $\mathit{items}(\mathit{matching}(S[\ell+1], \mathit{shift}(S, \ell+1)))$. Note that $\mathit{boundary}_R(S, \Pi, \ell, \mathit{offset})$ is the price of the item $V[\mathit{sum}(S, \ell) + \mathit{offset}]$ represented by Π, where V denote $\mathit{items}(S)$, and it can be computed in constant time.

Our efficient computation of a price-block sequence that belongs to $\mathit{prices}(\Pi, S, u^*, S')$ is shown in Algorithm 2.4. First, the algorithm performs a scan over the blocks of S' to compute a block index ℓ^* at line 6 so that each bid in each block $S[i]$ for $1 \leq i < \ell^*$ (resp., $\ell^* < i \leq |S|$) is less (resp., greater) than u^*. Then it is easy to see that the integer j^* computed at line 7 is equal to $\mathit{index}(u^*, U' + u^*)$, as in Algorithm 2.3.

Algorithm 2.4 Fast implementation of computing a price-block sequence that belongs to $\mathit{prices}(\Pi, S, u^*, S')$. Note: Continues on the next page.

```
Input: A price-block sequence $\Pi$, a superblock $S$, a bid $u^*$, and a superblock
    $S'$ such that $\mathit{prices}(\Pi, S, u^*, S')$ is well-defined.
Output: A price-block sequence $\Pi'$ such that the outcome $(S', \Pi')$ represents
    $\mathit{insert}(S, \Pi, u)$, where $u$ denotes $\mathit{bids}(S') \setminus \mathit{bids}(S)$.
 1: Let $U'$ denote $\mathit{bids}(S')$, let $V$ denote $\mathit{items}(S')$, and let $M'$ denote
    $\mathit{matching}(S')$
 2: Let $S'[i]$ be $(U'_i, V'_i)$ for $1 \leq i \leq |S'|$
 3: Let $p$ and $p'$ be two uninitialized arrays of $|M'|$ real values
 4: $\sigma(i) \leftarrow \mathit{sum}(S, i)$ for $0 \leq i \leq |S|$
 5: $\sigma'(i) \leftarrow \mathit{sum}(S', i)$ for $0 \leq i \leq |S'|$
 6: $\ell^* \leftarrow \max(1, |\{(U', V') \mid (U', V') \in S' \text{ and } U'[1] < u^*\}|)$
 7: $j \leftarrow j_L \leftarrow j^* \leftarrow \mathit{index}(u^*, U'_{\ell^*} + u^*) + \sigma'(\ell^* - 1)$
```

Algorithm 2.4, cont.

```
8: if $j^* > 1$ then
9:     $\ell \leftarrow \ell^*$
10:    $p[\sigma(\ell-1)+1 : \sigma(\ell)] \leftarrow \textit{explicit}(S, \Pi, \ell)$
11:    $p[\sigma'(\ell-1)+1] \leftarrow \textit{boundary}_L(S, \Pi, \ell, \sigma'(\ell-1) - \sigma(\ell-1))$
12:    $p[\sigma'(\ell)] \leftarrow \textit{boundary}_R(S, \Pi, \ell, \sigma'(\ell) - \sigma(\ell))$
13:    while $j > \sigma'(\ell-1)+1$ do
14:        Perform lines 8 through 14 of Algorithm 2.3
15:    end while
16:    if $j_L > \sigma'(\ell-1)+1$ then
17:        $p'[\sigma'(\ell-1)+1 : j_L - 1] \leftarrow p[\sigma'(\ell-1)+1 : j_L - 1]$
18:    else
19:        $\ell \leftarrow \ell - 1$
20:        while $\ell \geq 1$ do
21:            $t \leftarrow p'[j] - w(U'[j], V[j]) + w(U'[j], V[j-1])$
22:            $j_1 \leftarrow \sigma'(\ell-1)+1$
23:            $p[j_1] \leftarrow \textit{boundary}_L(S, \Pi, \ell, \sigma'(\ell-1) - \sigma(\ell-1))$
24:            if $p[j_1] \leq t + \Delta_L(M'[j_1 : j-1]) + w(U'[j_1], V[j_1])$ then
25:                $p'[j_1] \leftarrow t + \Delta_L(M'[j_1 : j-1]) + w(U'[j_1], V[j_1])$
26:                $\Pi'[\ell] \leftarrow (L, t)$
27:                $j \leftarrow j_L \leftarrow j_1$
28:                $\ell \leftarrow \ell - 1$
29:            else break
30:            end if
31:        end while
32:        if $\ell \geq 1$ then
33:            Perform lines 10 through 15 above
34:            $p'[\sigma'(\ell-1)+1 : j_L - 1] \leftarrow p[\sigma'(\ell-1)+1 : j_L - 1]$
35:            $\Pi'[\ell] \leftarrow p'[\sigma'(\ell-1)+1 : \sigma'(\ell)]$
36:        end if
37:    end if
38:    $\Pi'[\,:\ell-1] \leftarrow \Pi[\,:\ell-1]$
39: end if
40: Compute $j_R$ and the price-blocks for the items $V[j^*:\,]$ that represent the
    prices in $p^\dagger$, using code symmetric to lines 7 through 39
41: $\Pi'[\ell^*] \leftarrow p'[\sigma'(\ell^*-1)+1 : \sigma'(\ell^*)]$
42: return $\Pi'$
```

We compute $j_L^\dagger$ and the price-blocks that represent $p^\dagger$ for the items $V[\,:j^*-1]$ in three stages. The computation of $j_R^\dagger$ and the price-blocks for the items $V[j^*:\,]$ is symmetric and omitted. We first give some useful definitions that are analogous to those of P_L and P_R introduced in Section 2.7.5, and then we describe the three stages of Algorithm 2.4.

We define the state predicate Q_L (resp., Q_R) to hold if $j_L^\dagger \leq j_L$ (resp., $j_R^\dagger \geq j_R$) and for each item index j' such that $j_L \leq j' \leq j^*-1$ (resp., $j^* \leq j' \leq j_R$), either (1) $p'[j']$ is assigned $p^\dagger[j']$, or (2) $p'[j']$ remains uninitialized and the price of $V[j']$ represented by Π' is equal to $p^\dagger[j']$.

The first stage (lines 13 through 15) is identical to the left-scan of Algorithm 2.3 except that it is confined within the block $S'[\ell^*]$. The goal of the first stage is to fill $p'[\sigma'(\ell^*-1)+1 : j^*-1]$ which, together with $p'[j^* : \sigma'(\ell^*)]$ that is computed in the symmetric stage (i.e., the first stage associated with the computation of $j_R^\dagger$ and the price-blocks for the items $V[j^*:\,]$), constitutes the explicit price-block $\Pi'[\ell^*]$ built at line 41. Before executing the **while** loop at lines 13 through 15 that decrements j_L and fills p', we ensure at lines 10 through 12 that the input prices (prices represented by Π) for the items that are matched in $S'[\ell^*]$ (items in $V[\sigma'(\ell^*-1)+1 : \sigma'(\ell^*)]$) are available in the array p. Then, Lemma 2.7.14 below implies that $j_L = \max(j_L^\dagger, \sigma'(\ell^*-1)+1)$ and $p'[j_L : j^*-1] = \mathit{tight}_L(M'[j_L : j^*-1], w(u^*, V[j^*-1]))$ upon completion of the **while** loop. Hence, if the inequality checked at line 16 holds, we have $j_L = j_L^\dagger$, and we copy the unchanged prices for the remaining items into $p'[\sigma'(\ell^*-1)+1 : j_L-1]$ from $p[\sigma'(\ell^*-1)+1 : j_L-1]$ at line 17, and we are done. Otherwise, we proceed to the next stage to see whether $j_L^\dagger < \sigma'(\ell^* - 1) + 1$.

Lemma 2.7.14. The following claims hold: (1) the predicate Q_L is an invariant of the **while** loop at lines 13 through 15; (2) upon termination of the **while** loop at lines 13 through 15, if $j_L > \sigma'(\ell^* - 1) + 1)$ then $j_L = j_L^\dagger$.

Proof. The predicate Q_L trivially holds upon execution of line 7, and thus at the beginning of the first iteration of the **while** loop, by the definition of $j_L^\dagger$. The rest of the proof is identical to that of Lemma 2.7.10, with Q_L replacing P_L, except that if the **while** loop terminates because $j = \sigma'(\ell^* - 1) + 1$ at line 13, then Q_L only implies $j_L = j = \sigma'(\ell^* - 1) + 1 \geq j_L^\dagger$, but not necessarily $j_L = j_L^\dagger$. □

The second stage (lines 19 through 31) identifies the blocks that reside in the left-tight interval of $(M', p^\dagger)$ by utilizing the constant-time computation of Lemma 2.7.13. The prices corresponding to these blocks are represented using compact price-blocks. More precisely, for each block index i such that $1 \leq i < \ell^*$ and $j_L^\dagger \leq \sigma'(i)$, the **while** loop at lines 20 through 31 sets the price-block $\Pi'[i]$ corresponding to block $S'[i]$ to a compact price-block representing the prices in $p^\dagger$. Upon termination of the **while** loop, the program variable ℓ is equal to $\min\{i \mid 1 \leq i \leq \ell^* \text{ and } j_L^\dagger \leq \sigma'(i)\} - 1$. The following lemma establishes useful properties of the second stage.

Lemma 2.7.15. The following claims hold: (1) the predicate Q_L is an invariant of the **while** loop at lines 20 through 31; (2) upon termination of the

while loop at lines 20 through 31, we have $j_L = \sigma'(\ell) + 1$, and if $\ell \geq 1$ then $j_L^\dagger > \sigma'(\ell - 1) + 1$.

Proof. The predicate Q_L holds upon execution of line 19, and thus at the beginning of the first iteration of the **while** loop, by the first claim of Lemma 2.7.14. It is easy to see that the second claim holds if the loop never iterates, i.e., if $\ell = 0$ upon execution of line 19. We prove by induction on the number of iterations that Q_L holds at the end of each iteration. Consider an arbitrary iteration of the loop and assume that Q_L holds at the beginning of the iteration. The variable j_1 is set at line 22 to the index of the item that is matched to the leftmost bid in $S'[\ell]$; the following line sets $p[j_1]$ to the price of that item as represented by the input Π. Then, the algorithm determines whether $j_L^\dagger \leq j_1$ at line 24 by utilizing the constant-time computation of Lemma 2.7.13, where j plays the role of j_2: it is easy to see that the right-hand side of the condition of the **if** statement at line 24 is equal to $p'[j] + \Delta_L(M'[j_1 : j]) + w(U'[j_1], V[j_1])$ by (L1), and thus the condition holds if and only if $j_L^\dagger \leq j_1$. If $j_L^\dagger \leq j_1$, then Q_L and the definition of $p^\dagger$ imply that the price assigned at line 25 is equal to $p^\dagger[j_1]$, the price stored in the variable t is equal to $p^\dagger[j]$, and $p^\dagger[j_1 : j]$ is equal to $\mathit{tight}_L(M'[j_1 : j], t)$; thus, the predicate Q_L is maintained after setting $\Pi'[\ell]$ to the left-tight compact price-block at line 26 and upon executing line 28. If $j_L^\dagger > j_1$, it is easy to see that Q_L continues to hold and the second claim of the lemma holds. □

The third stage (line 33) is similar to the first stage: it is identical to the left-scan of Algorithm 2.3 except that it is confined within the block $S'[\ell]$. The goal of the third stage is to fill $p'[\sigma'(\ell-1)+1 : \sigma'(\ell)]$ which subsequently forms the explicit price-block $\Pi'[\ell]$ at line 35. Before executing the **while** loop that decrements j_L and fills p', we ensure that the input prices (prices represented by Π) for the items that are matched in $S'[\ell]$ (items in $V[\sigma'(\ell-1)+1 : \sigma'(\ell)]$) are available in the array p. By the second claim of Lemma 2.7.15 and by Lemma 2.7.16 below, it is easy to see that the **while** loop terminates with $j_L = j_L^\dagger$, and we have $p'[j_L : j^*-1] = \mathit{tight}_L(M'[j_L : j^*-1], w(u^*, V[j^*-1]))$.

Lemma 2.7.16. The following claims hold: (1) the predicate Q_L is an invariant of the **while** loop associated with line 33; (2) upon termination of the **while** loop associated with line 33, $j_L = j_L^\dagger$.

Proof. The predicate Q_L holds right before executing line 33, and thus it holds at the beginning of the first iteration of the **while** loop, by the first claim of Lemma 2.7.15. The rest of the proof is identical to that of Lemma 2.7.10, with Q_L replacing P_L, except that it is guaranteed by the second claim of Lemma 2.7.15 that the **while** loop terminates via the **break** statement, and thus, the inequality $j_L^\dagger > j = j_L - 1$ and Q_L imply that $j_L = j_L^\dagger$ holds upon termination of the loop. □

Corollary 2.7.17. Upon execution of line 39, $j_L = j_L^\dagger$.

Lemma 2.7.18. Algorithm 2.4 is correct.

Proof. It is sufficient to prove that the price-block sequence Π' represents the prices in $p^\dagger$. In what follows, let $\ell^\dagger$ denote $\min\{i \mid 1 \le i \le \ell^*$ and $j_L^\dagger \le$

$\sigma'(i)\} - 1$. As discussed earlier, upon termination of the **while** loop at lines 20 through 31, the program variable ℓ is equal to $\ell^\dagger$.

Lemma 2.7.14 and line 17 imply that $p'[\sigma'(\ell^* - 1) + 1 : j^* - 1] = p^\dagger[\sigma'(\ell^* - 1) + 1 : j^* - 1]$. The symmetric results and code associated with the computation of $j_R^\dagger$ and the price-blocks for the items $V[j^* :]$ imply that $p'[j^* : \sigma'(\ell^*)] = p^\dagger[j^* : \sigma'(\ell^*)]$. Then line 41 implies that $\Pi'[\ell^*]$ represents $p^\dagger[\sigma'(\ell^* - 1) + 1 : \sigma'(\ell^*)]$.

If $\ell^\dagger < \ell^* - 1$, then Lemma 2.7.15 and line 26 imply that $\Pi'[\ell^\dagger + 1 : \ell^* - 1]$ represents $p^\dagger[\sigma'(\ell^\dagger) + 1 : \sigma'(\ell^* - 1)]$.

If $\ell^\dagger \geq 1$, then Lemma 2.7.16 and lines 34 and 35 imply that $\Pi'[\ell^\dagger]$ represents $p^\dagger[\sigma'(\ell^\dagger - 1) + 1 : \sigma'(\ell^\dagger)]$.

If $\ell^\dagger > 1$, then Corollary 2.7.17 and line 38 imply that $\Pi'[: \ell^\dagger - 1]$ represents $p^\dagger[: \sigma'(\ell^\dagger - 1)]$, as those prices are unaffected by bid insertion.

If $\ell^* < |S'|$, the symmetric results and code associated with the computation of $j_R^\dagger$ and the price-blocks for the items $V[j^* :]$ imply that $\Pi'[\ell^* + 1 :]$ represents $p^\dagger[\sigma'(\ell^*) + 1 :]$. $\square$

It is easy to see that the three stages associated with the computation of $j_L^\dagger$ and the price-blocks that represent $p^\dagger$ for the items $V[: j^* - 1]$ run in $O(\sqrt{n})$ time where n denotes *size*(S): for the first and third stages, the number of iterations of each loop is at most the number of bids in the associated block, which is $O(\sqrt{n})$; for the second stage, the number of iterations of the loop is at most the number of blocks in S', which is $O(\sqrt{n})$; each iteration of each loop takes constant time; all the remaining operations can be performed in

$O(\sqrt{n})$ time. By a symmetric argument, the three stages associated with the computation of $j_R^\dagger$ and the price-blocks for the items $V[j^* :]$ also run in $O(\sqrt{n})$ time.

2.8 Concluding Remarks

We conclude this chapter with some remarks regarding the SOM. It is possible to support constant-time queries that return the bid matched to a given item with some additional bookkeeping. Queries to find whether a bid is matched or not, and if so, to return the matched item, can be implemented in logarithmic time by performing binary search. Finally, it is possible to initialize the SOM with a matching consisting of all dummy bids, each with intercept zero and slope zero, in linear time, since all of the weights involving those bids are zero, and thus it is trivial to construct the blocks.

Chapter 3

Computing VCG Prices Given a VCG Allocation of a UDALEW

This chapter provides our $O(n \log n)$-time algorithm for the second problem mentioned in Section 1.1.1, the problem of computing the VCG prices of a UDALEW, given a VCG allocation. An abbreviated version of the results presented in this chapter appears in a conference publication [15].

We first briefly describe our approach. Recall that, as discussed in Section 2.7.1, one characterization of the vector of VCG prices is that it is the minimum stable price vector. Thus a naive algorithm would start with zero prices and then look for and eliminate the instabilities (blocking pairs). While inspecting a particular blocking pair, the algorithm would increase the prices just enough to eliminate that pair. We take a similar approach, but with some additional care. We start with a minimum price vector that does not yield any blocking pairs involving an unassigned bid, by utilizing the geometric concept of the upper envelope. We then inspect the remaining blocking pairs using two scans of items. The first scan is in increasing order of item qualities, and the second scan is in decreasing order of item qualities. The most expensive step is the computation of the upper envelope, which takes $O(n \log n)$ time.

3.1 Algorithm 3.1

We now present our fast algorithm for computing the VCG prices given a VCG allocation (an MWM) of a UDALEW and then prove its correctness. Throughout this section, we fix a UDALEW $A = (U, V)$ and an MWM M of this UDALEW. Our algorithm takes A and M as input. To keep the presentation and the correctness argument simpler, we assume that the MWM M is an ordered matching. It is easy to see by Lemma 2.3.1 that any MWM that is not ordered can be converted to an ordered MWM by sorting the matched bids. We start with some useful definitions concerning the input UDALEW $A = (U, V)$.

Let M' be an MWM of A and let V' be the set of all items in V that are matched in M'. Then for any item v in V', we define the successor of v in M' as the item v' in V' with the smallest index that is still larger than that of v. Similarly for any item v in V', we define the predecessor of v in M' as the item v' in V' with the largest index that is still smaller than that of v. Note that no successor (resp., predecessor) exists for the item in V' with the largest (resp., smallest) index.

Algorithm 3.1 computes the minimum stable price vector in $O(n \log n)$

time given M and A. The high-level idea is to start with the minimum prices (**for** loop at lines 4–6) so that no unmatched bid can participate in a blocking pair (Lemma 3.1.4), then to raise the prices in a particular order only when necessary while maintaining the invariant that the price vector is at most the minimum stable price vector (Lemma 3.1.2). The price increases take place in two scans of the items, first in increasing and then in decreasing order of qualities. The prices are raised so that no matched item v can form a blocking pair with a bid that is matched to the predecessor or to the successor of v (Lemma 3.1.5). The latter result implies that no matched item can form a blocking pair with a matched bid (Lemma 3.1.6). We start with some lemmas, and then we establish the correctness and argue the performance of the algorithm in Theorem 3.1.8.

The following lemma is used in the proofs of Lemmas 3.1.2 and 3.1.4.

Lemma 3.1.1. Let v be an item in V. Let U' denote the set of all bids u such that $w(u,v) \geq 0$ and u is unmatched in M. If U' is empty, then $p(v)$ is 0 upon executing line 6. If U' is nonempty, then $p(v)$ is equal to $max_{u \in U'} w(u,v)$ upon executing line 6.

Proof. By the definition of the upper envelope. $\square$

Lemma 3.1.2. Let p^* denote the minimum stable price vector. Then for any item v, the price set by the algorithm for v at any point during the execution is at most $p^*(v)$.

Proof. Let s^* denote the surplus vector corresponding to p^*. Suppose that the claim is false, and consider the first time the price of some item v is set to a

Algorithm 3.1 An $O(n \log n)$-time algorithm for computing the minimum stable price vector given a VCG allocation of a UDALEW.

```
Input: A UDALEW $A = (U, V)$ and an ordered MWM $M$ of $A$.
Output: The minimum stable prive vector $p$ of $A$.
 1: Initialize $p(v)$ to 0 for all $v \in V$
 2: $L \leftarrow$ the set of lines corresponding to the bids that are not matched
    in $M$ (the line corresponding to a bid $u$ has slope $u.slope$ and intercept
    $u.intercept$)
 3: $H \leftarrow$ the upper envelope of $L$
 4: for all $v \in V$ do
 5:     $p(v) \leftarrow \max(p(v), H(v.quality))$
 6: end for
 7: $V' \leftarrow$ the set of all items in $V$ that are matched in $M$
 8: $v \leftarrow$ the item in $V'$ with the smallest index
 9: while $v \neq$ the item in $V'$ with the largest index do
10:     $v' \leftarrow$ the successor of $v$ in $M$
11:     $u \leftarrow$ the bid matched to $v$ in $M$
12:     $p(v') \leftarrow \max(p(v'), p(v) + u.slope \cdot (v'.quality - v.quality))$
13:     $v \leftarrow v'$
14: end while
15: $v' \leftarrow$ the item in $V'$ with the largest index
16: while $v' \neq$ the item in $V'$ with the smallest index do
17:     $v \leftarrow$ the predecessor of $v'$ in $M$
18:     $u' \leftarrow$ the bid matched to $v'$ in $M$
19:     $p(v) \leftarrow \max(p(v), p(v') - u'.slope \cdot (v'.quality - v.quality))$
20:     $v' \leftarrow v$
21: end while
22: return $p$
```

value, which we denote by $p(v)$, that is greater than $p^*(v)$. We consider two cases.

Case 1: The first occurrence is at line 5. Then Lemma 3.1.1 implies that there is an unmatched bid u such that $w(u, v) = p(v) > p^*(v)$. Since p^* is stable and since u is unmatched in M, we know $s^*(u) = 0$. But then

$s^*(u) + p^*(v) < w(u, v)$, contradicting the stability of p^*.

Case 2: The first occurrence is at line 12 (resp., line 19) and we denote the predecessor (resp., successor) of v by v'. Let u' be the bid that is matched to item v' in M. Then, immediately after the price of v is set to $p(v)$, we have $w(u', v) - p(v) = w(u', v') - p(v')$. Since this is the first time that the price for an item exceeds its minimum stable price, we know that $p(v') \leq p^*(v')$, and hence

$$w(u', v) - p(v) \geq w(u', v') - p^*(v').$$

But, since $p(v) > p^*(v)$, we have

$$w(u', v) - p^*(v) > w(u', v') - p^*(v'),$$

contradicting the stability of p^* since u' is matched to v' in M.

Hence the price of an item v is never increased beyond $p^*(v)$. □

Corollary 3.1.3. Let s^* denote the utility vector corresponding to the minimum stable prices. Then for any bid u, $s(u) \geq s^*(u)$ upon termination of the algorithm.

Proof. Follows from the definition of $s(u)$ and from Lemma 3.1.2. □

Lemma 3.1.4. Let u be a bid in U that is unmatched in M, and let v be an item in V. Then $s(u) + p(v) \geq w(u, v)$ upon termination of the algorithm.

Proof. Since u is not matched in M, we have $s(u) = 0$. Then the result follows from Lemma 3.1.1 and from the fact that prices never decrease. □

Lemma 3.1.5. Let v and v' be two items in V such that both v and v' are matched and v' is the successor of v in M. Let u denote the bid that is matched to v and let u' denote the bid that is matched to v'. Then $s(u)+p(v') \geq w(u, v')$ and $s(u') + p(v) \geq w(u', v)$ upon termination of the algorithm.

Proof. First consider the claim $s(u) + p(v') \geq w(u, v')$. The first **while** loop (lines 9 through 14) iterates through the items from lowest index to highest, so it ensures that

$$\begin{aligned} p(v') &\geq p(v) + u.\mathit{slope} \cdot (v'.\mathit{quality} - v.\mathit{quality}) \\ &= p(v) + w(u, v') - w(u, v) \\ &= w(u, v') - s(u) \end{aligned}$$

holds upon termination of the loop. The prices never decrease, so the claim remains satisfied after any change to the price of item v'. We now show that any increase to the price of item v does not violate the claim. Note that after the first **while** loop (lines 9 through 14) is terminated, such an increase can only happen once and it occurs in line 19. If it happens, the price of v is set to:

$$\begin{aligned} p(v) &= p(v') - u'.\mathit{slope} \cdot (v'.\mathit{quality} - v.\mathit{quality}) \\ &= p(v') - w(u', v') + w(u', v). \end{aligned} \tag{3.1}$$

If this increase violates the claim, then we have

$$w(u,v) - p(v) + p(v') < w(u,v'),$$

and substituting for $p(v)$ from (3.1) gives

$$w(u,v) + w(u',v') < w(u,v') + w(u',v),$$

which contradicts that M is an MWM because switching the items assigned to bids u and u' results in a heavier matching.

Now consider the claim $s(u') + p(v) \geq w(u',v)$. The second **while** loop (lines 16 through 21) iterates through the items from highest index to lowest, so it ensures that

$$\begin{aligned} p(v) &\geq p(v') - u'.\mathit{slope} \cdot (v'.\mathit{quality} - v.\mathit{quality}) \\ &= p(v') - w(u',v') + w(u',v) \\ &= w(u',v) - s(u') \end{aligned}$$

holds upon termination of the loop. □

Lemma 3.1.6. Let u be a bid in U and let v be an item in V such that both u and v are matched in M. Then $s(u) + p(v) \geq w(u,v)$ upon termination of the algorithm.

Proof. If v is matched to u in M, then $s(u) + p(v) = w(u,v)$ by definition. Suppose that u is matched to some item v' that is not equal to v. We give the proof for the case where the index of v' is smaller than that of v; the other case is symmetric. Since the index of v' is smaller than that of v, there exists a unique sequence $\langle v_1, \ldots, v_k \rangle$ of matched items in M such that $v_1 = v'$, $v_k = v$, and v_{i+1} is the successor of v_i in M for $1 \leq i < k$. Given such a sequence, let $P(i)$ denote the inequality $s(u) + p(v_i) \geq w(u, v_i)$. We show by induction that $P(k)$ holds. The base case $P(2)$ is implied by Lemma 3.1.5 since v_2 is the successor of $v_1 = v'$. Now assume that $P(i)$ holds for some i such that $2 \leq i < k$. Let u_i denote the bid to which item v_i is matched. Since v_{i+1} is the successor of v_i, Lemma 3.1.5 implies that

$$s(u_i) + p(v_{i+1}) \geq w(u_i, v_{i+1}).$$

Adding inequality $P(i)$ to the preceding inequality we obtain

$$s(u) + p(v_i) + s(u_i) + p(v_{i+1}) \geq w(u, v_i) + w(u_i, v_{i+1}),$$

and hence

$$s(u) + p(v_{i+1}) \geq w(u, v_i) + w(u_i, v_{i+1}) - w(u_i, v_i)$$

since $p(v_i) + s(u_i) = w(u_i, v_i)$.

By the definition of $w(u, v)$, we have

$$\begin{aligned} w(u, v_i) + w(u_i, v_{i+1}) - w(u_i, v_i) &= w(u, v_i) + u_i.\mathit{slope}(v_{i+1}.\mathit{quality} - v_i.\mathit{quality}) \\ &\geq w(u, v_i) + u.\mathit{slope}(v_{i+1}.\mathit{quality} - v_i.\mathit{quality}) \\ &= w(u, v_{i+1}), \end{aligned}$$

where the second line follows since $v_{i+1}.\mathit{quality} \geq v_i.\mathit{quality}$ and since the assumption that M is an ordered MWM of A implies $u_i.\mathit{slope} \geq u.\mathit{slope}$. Hence $P(i + 1)$ holds. $\square$

Lemma 3.1.7. Let u be a bid in U that is matched in M, and let v be an item in V that is not matched in M. Then $s(u) + p(v) \geq w(u, v)$ upon termination of the algorithm.

Proof. Let v' be the item that is matched to u. Let p^* denote the minimum stable price vector and let s^* denote the corresponding surplus vector. Then $s^*(u) + p^*(v) \geq w(u, v)$. Since v is unmatched in M we have $p^*(v) = 0$. Thus, Lemma 3.1.2 and the fact that the algorithm never sets a price to a negative value together imply that $p(v) = 0$. Corollary 3.1.3 implies $s(u) \geq s^*(u)$. Hence $s(u) \geq w(u, v)$, as required. $\square$

Theorem 3.1.8. The algorithm computes the minimum stable price vector in $O(n \log n)$ time.

Proof. Since the algorithm maintains nonnegative prices, no item blocks M. Corollary 3.1.3 implies that no bid blocks M. Lemmas 3.1.4, 3.1.6, and 3.1.7

together imply that no bid-item pair blocks M. By combining the preceding observations we deduce that the final price vector is stable. Hence Lemma 3.1.2 implies that the final price vector is the minimum stable price vector.

The most expensive step in the algorithm is the computation of the upper envelope of n lines. It is known that the lower convex hull of points and the upper envelope of lines are dual to each other [13, Section 11.4]. Since the lower convex hull can be computed in $O(n \log n)$ time [45, Theorem 3.7] and the two **while** loops take linear time, the complexity of the algorithm is $O(n \log n)$. $\square$

Chapter 4

UDALEWs with Evenly-Spaced Qualities and Applications to Scheduling

This chapter presents the three results mentioned in Section 1.1.2: Section 4.2 presents our $O(n \log n)$-time algorithm for solving Problem 1.1; Section 4.3 describes how to use the algorithm of Section 4.2 to solve Problem 1.2 in $O(n \log n)$ time; Section 4.4 shows that certain natural variations of Problem 1.1 are NP-hard. An abbreviated version of the results presented in this chapter appears in a conference publication [15].

As mentioned in Section 1.1.2, Problem 1.1 is equivalent to the problem of computing a VCG allocation of a UDALEW where the item qualities form an arithmetic sequence, and Problem 1.2 is equivalent to the problem of computing a VCG allocation of a UDALEW where the item qualities form a nondecreasing sequence that is the concatenation of two arithmetic sequences. Since our interest in these special cases is motivated by the applications to the scheduling domain, throughout this chapter, however, we deviate from the terminology of Chapter 2 and we adopt a terminology that is common in such applications.

We begin by revisiting Problems 1.1 and 1.2, and briefly reviewing related work. In both problems, we study scheduling unit jobs, i.e., jobs with an execution requirement of one time unit, with individual weights (w_i) and profits (e_i) on a single machine with a common deadline ($\overline{d}$) where some jobs may be rejected. If a job is scheduled by the deadline then its completion time is denoted by C_i; otherwise it is considered rejected. Let S denote the set of scheduled jobs and $\overline{S}$ denote the set of rejected jobs. In Problem 1.1, the goal is to minimize the sum of the weighted completion times of the scheduled jobs plus the total profits of the rejected jobs. Hence job profits can be equivalently interpreted as rejection penalties. We represent this problem using Graham's notation [32] as:

$$1 \mid p_i = 1, \overline{d}_i = \overline{d} \mid \sum_{S} w_i C_i + \sum_{\overline{S}} e_i \ . \qquad \text{(1.1 revisited)}$$

In Problem 1.2, we incorporate weighted tardiness penalties with respect to a common due date, i.e., every job also has a common due date d, and completing a job after the due date incurs an additional tardiness penalty that depends on its weight and a positive constant c, where the tardiness of a job is defined as $T_i = \max\{0, C_i - d\}$. We represent this problem using Graham's notation as:

$$1 \mid p_i = 1, d_i = d, \overline{d}_i = \overline{d} \mid \sum_S w_i C_i + c \sum_S w_i T_i + \sum_{\overline{S}} e_i \ . \qquad \text{(1.2 revisited)}$$

In both problems, we assume that the number of jobs is at least $\overline{d}$. If not, letting U_+ (resp., U_-) denoting the set of the jobs with nonnegative (resp., negative) weights, it is easy to observe that there exists a solution in which each job in U_+ (resp., U_-) that is not rejected is scheduled in one of the first $|U_+|$ (resp., last $|U_-|$) slots. Using this observation, we can solve the given instance by solving two smaller instances for which our assumption is satisfied.

4.1 Related Work

More general cases of Problem 1.1, almost all dealing with variable processing times, have been studied extensively. One of the earliest works that considers job specific profits and lateness penalties [59] reduces to our problem in the special case of setting all processing times to 1 and all due dates to 0. Two recent surveys review the research on various scheduling problems in which it is typically necessary to reject some of the jobs in order to achieve optimality [53, 58]. Epstein et al. [24] focus on unit jobs but consider only the online version of the problem.

Engels et al. [23] give a pseudo-polynomial-time dynamic programming algorithm for the same objective except that variable processing times are allowed and no deadline restriction is imposed. Engels et al. first show that the decision version of the problem is NP-complete and then give an FPTAS. They also remark that a running time of $O(n^2)$ can be achieved for the special case of unit processing times; our work improves this bound to $O(n \log n)$.

4.2 A Fast Algorithm for Problem 1.1

We encode an instance of Problem 1.1 as a weighted matching problem on a graph drawn from a certain family. Below we define this family, which we call $\mathcal{G}$, and we discuss how to express an instance of Problem 1.1 in terms of a graph in $\mathcal{G}$.

We define $\mathcal{G}$ as the family of all complete edge-weighted bipartite graphs $G = (U, V, w)$ such that the following conditions hold: $|U| \geq |V|$; each left vertex u in U has two associated integers $u.\mathit{intercept}$ and $u.\mathit{slope}$; the left vertices are indexed from 1 in non-decreasing order of slopes, breaking ties arbitrarily; right vertices are indexed from 1; the weight $w(u, v)$ of the edge between a left vertex u and a right vertex v is equal to $u.\mathit{intercept} + u.\mathit{slope} \cdot j$ where j denotes the index of v. Note that a graph $G = (U, V, w)$ in $\mathcal{G}$ admits an $O(|U|)$-space representation. It is easy to see that the family $\mathcal{G}$ is equivalent to the family of (graphs corresponding to the) all UDALEWs (modulo the order of vertices, and the bid and item *id*'s) such that the number of bids is at least the number of items and the qualities of the items form the arithmetic sequence 1 through the number of items.

Let I be an instance of Problem 1.1. The instance I consists of a set of n jobs to schedule, each with a profit and a weight, and a common deadline $\overline{d}$ where we assume that $n \geq \overline{d}$ as discussed in the beginning of this chapter. We encode the instance I as a graph $G = (U, V, w)$ in $\mathcal{G}$ such that the following conditions hold: $|U| = n$; $|V| = \overline{d}$; each left vertex represents a distinct job in I; each right vertex represents a time slot in which a job in I can be scheduled; for each job in I and the vertex u that represents that job, $u.intercept$ is equal to the profit of the job and $u.slope$ is equal to the negated weight of the job. It is easy to see by inspecting the objective of Problem 1.1 that minimizing the weighted sum of completion times is equivalent to maximizing the same expression with negated weights, and minimizing the sum of the profits of the rejected jobs is equivalent to maximizing the sum of the profits of the scheduled jobs. Hence, instance I of Problem 1.1 is equivalent to the problem of finding a maximum weight matching (MWM) of a graph $G = (U, V, w)$ in $\mathcal{G}$ that encodes I. Given this correspondence between the two problems, we refer to the left vertices (resp., right vertices) of a graph in $\mathcal{G}$ as *jobs* (resp., *slots*). The problem of computing an MWM of a graph $G = (U, V, w)$ in $\mathcal{G}$ can be reduced to the maximum weight maximum cardinality matching (MWMCM) problem by adding $|V|$ dummy jobs, each with intercept and slope zero, to obtain a graph that also belongs to $\mathcal{G}$.

As a result of the equivalence of the two problems mentioned above and the reduction from the MWM to the MWMCM problem, we can obtain an

$O(n \log n)$-time algorithm for Problem 1.1 by providing an $O(|U| \log |U|)$-time algorithm to compute an MWMCM of a graph $G = (U, V, w)$ in $\mathcal{G}$. Before discussing this algorithm further, we introduce some useful definitions.

Let $G = (U, V, w)$ be a graph in $\mathcal{G}$. We say that a subset U' of U is *optimal* for G if there exists an MWMCM M of G such that the set of jobs that are matched in M is equal to U'. Lemma 4.2.1 below shows that it is straightforward to efficiently construct an MWMCM of G given an optimal set of jobs for G. Let U' be a subset of U with size $|V|$ and let $i_1 < \cdots < i_{|V|}$ denote the indices of the jobs in U'. Then we define *matching*(U') as the set of $|V|$ job-slot pairs obtained by pairing the job with index i_k to the slot with index k for $1 \leq k \leq |V|$. The following lemma is analogous to Lemma 2.3.1 and it is a straightforward application of the rearrangement inequality [34, Section 10.2, Theorem 368] to our setting.

Lemma 4.2.1. Let $G = (U, V, w)$ be a graph in $\mathcal{G}$. Let U' be a subset of U with size $|V|$. Let W denote the maximum weight of any MCM of G that matches U'. Then *matching*(U') is of weight W.

Having established Lemma 4.2.1, it remains to show how to efficiently identify an optimal set of jobs for a given graph $G = (U, V, w)$ in $\mathcal{G}$. The main technical result of this section is an $O(|U| \log |U|)$-time dynamic programming algorithm for accomplishing this task. The following definitions are useful for describing our dynamic programming framework.

Let $G = (U, V, w)$ be a graph in $\mathcal{G}$. For any integer i such that $0 \leq i \leq |U|$, we define U_i as the set of jobs with indices 1 through i. Similarly, for any integer j such that $0 \leq j \leq |V|$, we define V_j as the set of slots with indices 1 through j. For any integers i and j such that $0 \leq j \leq i \leq |U|$ and $j \leq |V|$, we define $G_{i,j}$ as the subgraph of G induced by the vertices $U_i \cup V_j$, and we define $W(i,j)$ as the weight of an MWMCM of $G_{i,j}$. Note that any subgraph $G_{i,j}$ of G also belongs to $\mathcal{G}$.

Let us define $\mathcal{G}^*$ as the family of all graphs in $\mathcal{G}$ having an equal number of slots and jobs. Given a graph $G = (U, V, w)$ in $\mathcal{G}^*$, our dynamic programming algorithm computes in $O(|U| \log |U|)$ total time an optimal set of jobs for each $G_{|U|,j}$ for $1 \leq j \leq |U|$. For any graph $G' = (U, V', w')$ in $\mathcal{G}$, we can construct a graph $G = (U, V, w)$ in $\mathcal{G}^*$ satisfying ${G'}_{|U|,j} = G_{|U|,j}$ for all $1 \leq j \leq |V'|$ by defining V as the set of $|U|$ slots indexed from 1 through $|U|$. Thus, given any graph $G' = (U, V', w')$ in $\mathcal{G}$, our algorithm can be used to identify an optimal set of jobs for each subgraph ${G'}_{|U|,j}$ for $1 \leq j \leq |V'|$ in $O(|U| \log |U|)$ total time.

Throughout the remainder of this section, we fix a graph instance $G = (U, V, w)$ in $\mathcal{G}^*$. The presentation of the algorithm is organized as follows. Section 4.2.1 introduces the core concept, which we call the *acceptance order*, that our algorithm is built on. Section 4.2.2 presents the key idea (Lemma 4.2.7) underlying our algorithm for computing the acceptance order. Finally, Section 4.2.3 describes an efficient augmented binary search tree implementation of the algorithm.

4.2.1 Acceptance Orders

Lemma 4.2.1 reduces Problem 1.1 to the problem of identifying an optimal subset of U for G. In addition to an optimal set of jobs for G, our algorithm determines for each integer i and j such that $0 \leq j \leq i \leq |U|$, a subset $\mathit{best}(i, j)$ of U_i that is optimal for $G_{i,j}$ (Lemma 4.2.5). There are quadratically many such sets, so in order to run in quasilinear time, we compute a compact representation of those sets by exploiting the following two properties. The first property is that $\mathit{best}(i, j-1)$ is a subset of $\mathit{best}(i, j)$ for $1 \leq j \leq i \leq |U|$. Thus, for a fixed i, the sequence of sets $\mathit{best}(i, 1), \ldots, \mathit{best}(i, i)$ induces an ordering σ_i of jobs U_i, which we later define as the acceptance order of U_i, where the job at position j of σ_i is the one that is present in $\mathit{best}(i, j)$ but not in $\mathit{best}(i, j-1)$. The second property is that σ_{i-1} is a subsequence of σ_i for $1 \leq i \leq |U|$. This second property suggests an incremental computation of σ_i's which will be exploited to find the weights of MWMCMs for all prefixes of jobs to solve Problem 1.2, as described in Section 4.3.

We now give the formal definitions of the acceptance order and the optimal set $\mathit{best}(i, j)$, and present two associated lemmas.

We say that a vertex is *essential* for an edge-weighted bipartite graph G if it belongs to every MWMCM of G.

For any integer i such that $0 \leq i \leq |U|$ we define σ_i inductively as follows: σ_0 is the empty sequence; for $i > 0$ let u denote the job with index i, then σ_i is obtained from σ_{i-1} by inserting job u immediately after the prefix of σ_{i-1} of length $p - 1$ where p, which we call the position of u in σ_i, is the minimum positive integer such that job u is essential for $G_{i,p}$. It is easy to see that σ_i is a sequence of length i and that $1 \leq p \leq i$ since u is trivially essential for $G_{i,i}$. Furthermore, σ_{i-1} is a subsequence of σ_i for $1 \leq i \leq |U|$, as claimed above.

We say that σ_i is the *acceptance order* of the set of jobs U_i. Note that $\sigma_{|U|}$ is the acceptance order of the set of all jobs.

The following lemma is used in the proof of Lemma 4.2.3.

Lemma 4.2.2. Let $G = (U, V, E)$ be an edge-weighted bipartite graph, let V' be a subset of V, let G' be the subgraph of G induced by the set of vertices $U \cup V'$, and let u be a vertex in U that is essential for G'. Then u is essential for G.

Proof. Assume that the claim is false, and let M be an MWMCM of G such that u is not matched in M. Let M' be an MWMCM of G'. Since u is essential for G', vertex u is matched in M'. Let G'' denote the graph $(U, V, M \oplus M')$. Since u is matched in M' and not in M, we deduce that u has degree one in G''. Since no vertex in G'' has degree more than two, we conclude that the connected component of G'' that includes u is a path, call it P, and that u is an endpoint of P. The edges of P alternate between M and M'; let X denote $P \cap M$ and let X' denote $P \cap M'$.

Case 1: P is of odd length. Since u is an endpoint of P and is u is matched in M', we deduce that $|X'| = |X| + 1$. It follows that $(M \setminus X) \cup X'$ is a matching of G with cardinality one higher than that of M, contradicting our assumption that M is an MWMCM of G.

Case 2: P is of even length. Thus $|X| = |X'|$ and every vertex in V that belongs to P is matched in both M and M', and hence belongs to V'. It follows that $(M' \setminus X') \cup X$ is an MCM of G'; in what follows, we refer to this MCM of G' as M''. Let W denote the total weight of the edges in X and let W' denote the total weight of the edges in X'.

Case 2.1: $W < W'$. Thus $(M \setminus X) \cup X'$ is an MCM of G with weight higher than that of M, contradicting our assumption that M is an MWMCM of G.

Case 2.2: $W > W'$. Thus M'' is an MCM of G' with weight higher than that of M', contradicting our assumption that M' is an MWMCM of G'.

Case 2.3: $W = W'$. Thus M'' is an MWMCM of G that does not match u, contradicting our assumption that u is essential for G'. $\square$

Lemma 4.2.3. Let i and j be any integers such that $1 \leq j \leq i \leq |U|$ and let u denote the job with index i. Then job u is essential for $G_{i,j}$ if and only if the position of u in σ_i is at most j.

Proof. Immediate from Lemma 4.2.2. $\square$

For any integers i and j such that $0 \leq j \leq i \leq |U|$, we define $\mathit{best}(i, j)$ as the set of the first j jobs in σ_i. Thus, $\mathit{best}(i, j-1)$ is a subset of $\mathit{best}(i, j)$ for $1 \leq j \leq i \leq |U|$, as claimed above.

The following lemma is used in the proof of Lemma 4.2.5.

Lemma 4.2.4. For any integers i and j such that $1 \leq j \leq i \leq |U|$, we have

$$best(i, j) = \begin{cases} best(i-1, j-1) + u & \text{if job } u \text{ is essential for } G_{i,j} \\ best(i-1, j) & \text{otherwise,} \end{cases}$$

where u denotes the job with index i.

Proof. Immediate from the definition of *best*(i, j) and from Lemma 4.2.3. □

Lemma 4.2.5. Let i and j be any integers such that $0 \leq j \leq i \leq |U|$. Then *matching*(*best*(i, j)) is an MWMCM of $G_{i,j}$.

Proof. For any integer i such that $0 \leq i \leq |U|$, let $P(i)$ denote the predicate "for any integer j such that $0 \leq j \leq i$, *matching*(*best*(i, j)) is an MWMCM of $G_{i,j}$". We prove by induction on i that $P(i)$ holds for all i such that $0 \leq i \leq |U|$. It is easy to see that $P(0)$ holds. Let i be an integer such that $0 < i \leq |U|$, and let u denote the job with index i. We now complete the proof by arguing that $P(i-1)$ implies $P(i)$. It is easy to see that *matching*(*best*$(i, 0)$) is an MWMCM of $G_{i,0}$. Thus in the remainder of the argument we may assume that j is an integer such that $1 \leq j \leq i$. We consider two cases.

Case 1: Job u is essential for $G_{i,j}$. Let v denote the slot with index j. By Lemma 4.2.1, and since the index of job u is greater than that of any job in $G_{i-1,j-1}$, we can obtain an MWMCM of $G_{i,j}$ by adding the edge (u, v) to an arbitrary MWMCM of $G_{i-1,j-1}$. By the induction hypothesis, *matching*(*best*$(i-1, j-1)$) is an MWMCM of $G_{i-1,j-1}$. Thus the matching obtained by adding edge (u, v) to *matching*(*best*$(i-1, j-1)$) is an MWMCM of

$G_{i,j}$. Since Lemma 4.2.4 implies that $\mathit{best}(i,j)$ is equal to $\mathit{best}(i-1,j-1)+u$, and since the index of job u is greater than that of any job in $\mathit{best}(i-1,j-1)$, the latter matching is equal to $\mathit{matching}(\mathit{best}(i,j))$. We conclude that $\mathit{matching}(\mathit{best}(i,j))$ is an MWMCM of $G_{i,j}$, as required.

Case 2: Job u is not essential for $G_{i,j}$. Thus any MWMCM of $G_{i-1,j}$ is an MWMCM of $G_{i,j}$. By the induction hypothesis, we conclude that $\mathit{matching}(\mathit{best}(i-1,j))$ is an MWMCM of $G_{i,j}$. Since Lemma 4.2.4 implies that $\mathit{best}(i,j)$ is equal to $\mathit{best}(i-1,j)$, we conclude that $\mathit{matching}(\mathit{best}(i,j))$ is an MWMCM of $G_{i,j}$, as required. $\square$

Lemmas 4.2.1 and 4.2.5 imply that once we compute the acceptance order $\sigma_{|U|}$, we can sort its first $\overline{d}$ jobs by their indices to obtain a matching to solve Problem 1.1.

4.2.2 Computing the Acceptance Order

As we have established the importance of the acceptance order $\sigma_{|U|}$, we now describe how to compute it efficiently. We start with σ_1 and introduce the tasks one by one in index order to compute the sequences $\sigma_2, \ldots, \sigma_{|U|}$ incrementally. Once we know σ_{i-1}, we just need to find out where to insert the job with index i in order to compute σ_i. We first introduce some definitions and a lemma, and then we describe the key idea (Lemma 4.2.7) for finding the position of a job in the corresponding acceptance order.

For any integers i and j such that $1 \leq j \leq i \leq |U|$, let $\sigma_i[j]$ denote the job with position j in σ_i, where $\sigma_i[1]$ is the first job in σ_i.

For any job u that belongs to U, we define *better*(u) as the set of jobs that precede u in σ_i where i denotes the index of u. Thus $|better(u)| = p - 1$ where p is the position of u in σ_i. The set *better*(u) is the set of jobs that precede u both in index order and in acceptance order.

Lemma 4.2.6. Let i and j be integers such that $1 \leq j \leq i \leq |U|$, and let i' denote the index of job $\sigma_i[j]$. Then the set of jobs in *best*$(i, j-1)$ with indices less than i' is equal to *better*$(\sigma_i[j])$.

Proof. Every job in *best*(i, j) has index at most i. Since $\sigma_i[j]$ belongs to *best*(i, j), we conclude that $i' \leq i$. Let p denote the position of $\sigma_i[j]$ in $\sigma_{i'}$. By definition, job $\sigma_i[j]$ occurs in position j of σ_i. By the definition of $\sigma_{i'}$, job $\sigma_i[j]$ occurs in position p of $\sigma_{i'}$. Since $\sigma_{i'}$ is a subsequence of σ_i, we conclude that *best*$(i, j-1)$ consists of *better*$(\sigma_i[j])$ plus $j - p$ additional jobs, each with index greater than i'. Since each job in *better*$(\sigma_i[j])$ has index less than i', the claim follows. $\square$

For any subset U' of U, we define *sum*(U') as $\sum_{u \in U'} u.slope$.

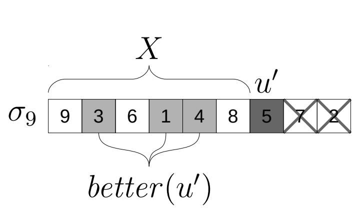
Now we are ready to discuss the idea behind the efficient computation of the acceptance orders incrementally. Assume that we already know the acceptance order σ_{i-1} of the set of the first $i - 1$ jobs for some integer i such that $1 < i \leq |U|$. Let u denote the job with index i. If we can determine in constant time, for any job in the set U_{i-1}, whether u precedes that job in σ_i, then we can perform a binary search in order to find in logarithmic time the position of u in σ_i. Suppose that we would like to know whether u precedes $\sigma_{i-1}[j]$ in σ_i for some integer j such that $1 \leq j < i$. In other words we would like to determine whether the position of u in σ_i is at most j. In what follows, let u' denote the job $\sigma_{i-1}[j]$ and let v denote the slot with index j. Then by Lemma 4.2.3, job u precedes u' in σ_i if and only if u is essential for $G_{i,j}$.

In order to determine whether job u is essential for $G_{i,j}$, we need to compare the weight of a heaviest possible matching for $G_{i,j}$ that does not include u to the weight of a heaviest possible matching for $G_{i,j}$ that includes u. The former weight is $W(i-1, j)$. Since job u has the highest index among the jobs with indices 1 through i, by Lemma 4.2.1, the latter weight is equal to $w(u, v) + W(i-1, j-1)$.

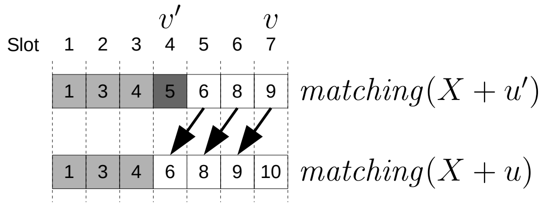
Let X denote $\mathit{best}(i-1, j-1)$. Since $\mathit{best}(i-1, j-1)+u' = \mathit{best}(i-1, j)$, Lemma 4.2.5 implies that the weight of $\mathit{matching}(X+u')$ is equal to $W(i-1, j)$. By Lemma 4.2.5, the weight of $\mathit{matching}(X)$ is $W(i-1, j-1)$. Since job u has the highest index among the jobs in $X+u$, the weight of $\mathit{matching}(X+u)$ is $w(u, v) + W(i-1, j-1)$.

Combining the results of the preceding paragraphs, we conclude that job u is essential for $G_{i,j}$ if and only if the weight of $\mathit{matching}(X+u)$ is greater than the weight of $\mathit{matching}(X+u')$.

Figure 4.1 shows an example where $i = 10$ and $j = 7$. Thus we are trying to determine whether the job with index 10 precedes $\sigma_9[7]$ in σ_{10}. In this example, u denotes the job with index 10 and u' denotes $\sigma_9[7]$, which is the job with index 5, as shown in Figure 4.1a. The set X is the first 6 jobs in σ_9. The jobs appearing past u' in σ_9, jobs with indices 7 and 2, do not participate in the matchings that we are interested in so they are crossed out.



(a) Acceptance order σ_9



(b) The two matchings to compare

Figure 4.1: An example in which we try to determine whether the job with index 10 precedes $\sigma_9[7]$ in σ_{10}. Each box represents the job whose index is shown inside.

Figure 4.1b shows the two matchings $matching(X + u')$ and $matching(X + u)$ of which we would like to compare the weights. As seen in Figure 4.1b, each job in X with index less than that of job u', shaded light gray in the figure, is matched to the same slot in both $matching(X + u)$ and $matching(X + u')$. By Lemma 4.2.6, those jobs are the ones in the set $better(u')$, which are the jobs with indices 1, 3 and 4 in the example. Hence job u' occurs in position $|better(u')| + 1$ when we sort the set of jobs $X + u'$ by index and thus it is matched to the slot with index $|better(u')|+1$ in $matching(X+u')$. Moreover, each job in X with index greater than that of job u' is matched to a slot with index one lower in $matching(X + u)$ than in $matching(X + u')$, as depicted by the arrows in Figure 4.1b for the jobs with indices 6, 8, and 9.

Hence the weight of $matching(X+u)$ minus the weight of $matching(X+u')$ is equal to $w(u, v) - w(u', v')$ plus the sum of the slopes of all jobs in $best(i - 1, j - 1)$ with indices greater than that of u', where v' denotes the slot with index $|better(u')| + 1$. By Lemma 4.2.6, the latter sum is equal to $sum(best(i - 1, j - 1)) - sum(better(u'))$. These observations establish the proof of the following lemma which we utilize in computing the acceptance orders incrementally.

Lemma 4.2.7. Let i and j be integers such that $1 \leq j < i \leq |U|$. Let u denote the job with index i and let u' denote the job $\sigma_{i-1}[j]$. Then the following are equivalent: (1) The position of u in σ_i is at most j; (2) Job u is essential for $G_{i,j}$; (3) The weight of $\mathit{matching}(\mathit{best}(i-1, j-1) + u)$ is greater than the weight of $\mathit{matching}(\mathit{best}(i-1, j-1) + u')$; (4) $w(u, v) > w(u', v') + \mathit{sum}(\mathit{best}(i-1, j-1)) - \mathit{sum}(\mathit{better}(u'))$ where v denotes the slot with index j and v' denotes the slot with index $|\mathit{better}(u')| + 1$.

4.2.3 Binary Search Tree Implementation

We employ a self-balancing binary search tree (BST) that stores the jobs in a certain order: an inorder traversal of the BST yields the acceptance order. We obtain an efficient algorithm for incrementally computing the acceptance orders by augmenting this BST so that we can apply Lemma 4.2.7 in constant time. The algorithm runs $|U|$ iterations where the job with index i is inserted into the BST at iteration i to obtain σ_i from σ_{i-1} by performing a binary search. We first give some definitions that are useful in the description of the algorithm and then we state in Lemma 4.2.8 how to perform the comparisons for the binary search.

For a binary tree T and an integer i such that $1 \leq i \leq |U|$, we define the predicate $\mathit{ordered}(T, i)$ to hold if T contains i nodes that represent the jobs U_i, and the sequence of the associated jobs resulting from an inorder traversal of T is σ_i. The job represented by a node x is denoted by $x.job$.

Let T be a binary tree satisfying $ordered(T, i)$ for some i. For any node x in T, $precede(x, T)$ is defined as the set of jobs associated with the nodes that precede x in an inorder traversal of T.

Lemma 4.2.8. Let i be an integer such that $1 < i \leq |U|$ and let u denote the job with index i. Let T be a binary tree satisfying $ordered(T, i-1)$ and let x be a node in T. Assume that $|precede(x, T)|$, $sum(precede(x, T))$, $|better(x.job)|$, and $sum(better(x.job))$ are given. Then we can determine in constant time whether u precedes $x.job$ in σ_i.

Proof. Let j denote $|precede(x, T)| + 1$. Then $ordered(T, i-1)$ implies that $x.job$ is $\sigma_{i-1}[j]$ and $sum(precede(x, T))$ is equal to $sum(best(i-1, j-1))$. Now let u' denote $\sigma_{i-1}[j]$. Then we can test Inequality 4 of Lemma 4.2.7 in constant time to determine whether the position of u in σ_i is at most j, thus whether u precedes u' in σ_i. □

Lemma 4.2.8 implies that once we know certain quantities about a node x in the BST then we can tell in constant time whether the new job precedes $x.job$ in the acceptance order. The necessary information to compute the first two of those quantities can be maintained by standard BST augmentation techniques as described in [11, Chapter 14]. The other two quantities turn out to be equal to the first two at the time the node is inserted into the BST and they can be stored along with the node. In order to present the details of our algorithm and how we augment the BST, we introduce the following useful definitions.

We define $\mathcal{T}$ as the set of all binary trees T satisfying the following two conditions: (1) the predicate $\mathit{ordered}(T, i)$ holds for some i; (2) each node x in T has integer fields $x.\mathit{size}$, $x.\mathit{sum}$, $x.\mathit{sizeLeft}$, and $x.\mathit{sumLeft}$ (in addition to the field $x.\mathit{job}$ implied by condition (1)).

For any tree T in $\mathcal{T}$ and any node x in T, we define $\mathit{subtree}(x)$ as the set of all jobs that are associated with the nodes in the subtree rooted at x. For convenience we define $\mathit{subtree}(\mathit{nil})$ as the empty set.

For any tree T in $\mathcal{T}$ and any node x in T, we define the predicate $\mathit{augmented}(x)$ to hold if the field $x.\mathit{size}$ is equal to $|\mathit{subtree}(x)|$, and the field $x.\mathit{sum}$ is equal to $\mathit{sum}(\mathit{subtree}(x))$. We refer to $x.\mathit{size}$ and $x.\mathit{sum}$ together as the *augmented fields* of node x.

For any tree T in $\mathcal{T}$ and any node x in T, we define the predicate $\mathit{historical}(x)$ to hold if the field $x.\mathit{sizeLeft}$ is equal to $|\mathit{better}(x.\mathit{job})|$, and the field $x.\mathit{sumLeft}$ is equal to $\mathit{sum}(\mathit{better}(x.\mathit{job}))$. We refer to $x.\mathit{sizeLeft}$ and $x.\mathit{sumLeft}$ together as the *historical fields* of node x.

For any tree T in $\mathcal{T}$ and any integer i, we define the predicate $\mathit{represent}(T, i)$ to hold if $\mathit{ordered}(T, i)$ holds, and for every node x in T, both $\mathit{augmented}(x)$ and $\mathit{historical}(x)$ hold.

Our algorithm grows an augmented BST by iteratively inserting each of the $|U|$ jobs, in index order. A concise implementation is given in Algorithm 4.1. The outer **while** loop (lines 3–32) runs for $|U|$ iterations. Each iteration i inserts the job with index i, referenced by the variable u, into the BST, whose root is pointed by the variable x at the beginning of the iteration. The inner **while** loop (lines 8–18) performs the binary search by starting from the root and descending to the insertion position of the new leaf representing the job u. When the execution reaches line 14, the variable *sizeLeft*$'$ is equal to $|\mathit{precede}(x, T)|$ and the variable *sumLeft*$'$ is equal to $\mathit{sum}(\mathit{precede}(x, T))$. The variable *flag* is set depending on the outcome of the application of Lemma 4.2.8.

Lines 19–30 attach the new leaf y to the BST and set its historical fields. The Fixup routine, which is called at line 31, fixes the augmented fields of the nodes along the path from the new leaf to the root, rebalances the BST using the standard techniques of the associated self-balancing BST implementation, and returns the root of the BST. (For a red-black tree implementation, the Fixup routine can be implemented as the RB-Insert-Fixup operation [11, Chapter 13] with the addition of setting the new node's color to red at the beginning, following the guidelines in [11, Chapter 14] for the augmentation.) Thus, at the end of each iteration of the outer **while** loop, the BST stores the jobs inserted so far in their acceptance order.

Theorem 4.2.9. Algorithm 4.1 computes the acceptance order of U in $O(|U| \log |U|)$ time.

Proof. Let T_i denote the BST obtained by the algorithm after i iterations, $1 \leq i \leq |U|$. We establish below that $\mathit{represent}(T_i, i)$ holds for $1 \leq i \leq |U|$. Given this claim, it is easy to verify that the historical fields of a node x remain constant after the initial creation of x. Thus the historical fields could just as easily be maintained in an array outside of the tree. In contrast, the

Algorithm 4.1 An efficient algorithm for incrementally computing the acceptance order by growing an augmented BST.

Input: A set U of jobs (sorted in non-decreasing order of slopes).

```
 1: $x \leftarrow nil$
 2: $i \leftarrow 0$
 3: while $i < |U|$ do
 4:     $i \leftarrow i + 1$
 5:     $u \leftarrow$ the job from the set $U$ with index $i$
 6:     $sizeLeft, sumLeft \leftarrow 0$
 7:     $x' \leftarrow x$
 8:     while $x' \neq nil$ do
 9:         $x \leftarrow x'$
10:         $sizeLeft' \leftarrow$ if $x.left = nil$ then $sizeLeft$ else $sizeLeft + x.left.size$
11:         $sumLeft' \leftarrow$ if $x.left = nil$ then $sumLeft$ else $sumLeft + x.left.sum$
12:         $v \leftarrow$ the slot with index $sizeLeft' + 1$
13:         $v' \leftarrow$ the slot with index $x.sizeLeft + 1$
14:         $flag \leftarrow w(u, v) \leq w(x.job, v') + sumLeft' - x.sumLeft$
15:         $x' \leftarrow$ if $flag$ then $x.right$ else $x.left$
16:         $sizeLeft \leftarrow$ if $flag$ then $sizeLeft' + 1$ else $sizeLeft$
17:         $sumLeft \leftarrow$ if $flag$ then $sumLeft' + x.job.slope$ else $sumLeft$
18:     end while
19:     $y \leftarrow$ a new tree node
20:     $y.job \leftarrow u$
21:     $y.sizeLeft \leftarrow sizeLeft$
22:     $y.sumLeft \leftarrow sumLeft$
23:     $y.parent \leftarrow x$
24:     if $x \neq nil$ then
25:         if $flag$ then
26:             $x.right \leftarrow y$
27:         else
28:             $x.left \leftarrow y$
29:         end if
30:     end if
31:     $x \leftarrow$ FIXUP$(y)$
32: end while
```

augmented fields form an integral part of the BST data structure, and may be updated as the structure of the tree changes.

We use a red-black tree to obtain the desired time bound. Cormen et al. [11, Chapters 13 and 14] present an implementation for red-black trees that rebalances the tree in logarithmic time while maintaining the augmented fields.

First we state an observation that will be utilized in the constant time comparison of the nodes and in the computation of the historical fields. Let T be a tree in $\mathcal{T}$ such that each node x in T satisfies $\mathit{augmented}(x)$. Then observe that we can compute $|\mathit{precede}(r, T)|$ and $\mathit{sum}(\mathit{precede}(r, T))$ in constant time where r denotes the root of T. Moreover, for each non-root node x in T, $|\mathit{precede}(x, T)|$ and $\mathit{sum}(\mathit{precede}(x, T))$ can be computed in constant time given $|\mathit{precede}(y, T)|$ and $\mathit{sum}(\mathit{precede}(y, T))$ where y denotes the parent of x.

The first iteration of the algorithm creates a BST T_1 with a single node x representing the job with index 1. The two historical fields of x are each initialized to zero, and the augmented fields are initialized as follows: $x.\mathit{size}$ is set to 1; $x.\mathit{sum}$ is set to the slope of the job with index 1. It is easy to verify that the predicate $\mathit{represent}(T_1, 1)$ holds.

We now describe a non-first iteration i, $1 < i \leq |U|$, where we assume inductively that $\mathit{represent}(T_{i-1}, i-1)$ holds. Let u denote the job with index i. In order to obtain T_i from T_{i-1}, we first insert a new leaf representing the job u into T_{i-1} without changing the structure of T_{i-1} other than linking the new leaf such that the resulting BST, call it T'_{i-1}, satisfies $\mathit{ordered}(T'_{i-1}, i)$. Note that there is a unique such BST T'_{i-1} since there is a unique position in the tree such that the new node representing u can be inserted without changing the edges between the existing nodes. This insertion position can be found by first comparing the job u with the job represented by the root of T_{i-1} (applying Lemma 4.2.8) and descending either left or right depending on whether u precedes the root in σ_i, thus following a path that terminates at the insertion position by comparing u with the jobs represented by the nodes on that path as in a standard BST insertion operation. By the observation in the previous paragraph, it is easy to see that $|\mathit{precede}(x, T_{i-1})|$ and $\mathit{sum}(\mathit{precede}(x, T_{i-1}))$ can be computed in constant time for each node x on the path that the binary search follows. Thus, together with the historical fields, we have the information to apply Lemma 4.2.8 in constant time for each comparison. Let y denote the new leaf that represents the job u. Again by the observation in the previous paragraph, since the last node that we compare u with is the parent of y, $|\mathit{precede}(y, T'_{i-1})|$ and $\mathit{sum}(\mathit{precede}(y, T'_{i-1}))$ can also be computed in constant time. Since $\mathit{ordered}(T'_{i-1}, i)$ implies $\mathit{better}(y.\mathit{job})$ is equal to $\mathit{precede}(y, T'_{i-1})$, the historical fields of y can be computed in constant time.

The process described above, which attaches the new leaf y representing the job with index i to T_{i-1} to obtain T'_{i-1}, takes time proportional to the depth of y in T'_{i-1}. Once y is added, we can update the augmented fields of the nodes on the path from y to the root within the same time bound so that $\mathit{augmented}(x)$ holds for each node x on that path. Note that the augmented fields of the other nodes are not affected by the insertion, thus $\mathit{represent}(T'_{i-1}, i)$

holds.

The final step at iteration i is to rebalance T'_{i-1} with the RB-INSERT-FIXUP operation [11, Chapters 13 and 14] in logarithmic time, while maintaining the augmented fields, in order to obtain $O(|U|\log|U|)$ overall running time. Let T_i denote the result of the rebalancing operation on T'_{i-1}. Since the augmented fields are maintained by the rebalancing operation, the BST T_i satisfies $\mathit{represent}(T_i, i)$ and the algorithm proceeds to the next iteration. Note that it is easy to argue the same performance for certain other balanced BST structures, e.g., $O(|U|\log|U|)$ amortized time for splay trees [57] where each insertion, together with the associated splay operation at the end of each iteration, takes amortized logarithmic time. $\square$

As mentioned earlier, once $\sigma_{|U|}$ is computed, we can extract an MWMCM of $G_{|U|,j}$ for any j such that $1 \leq j \leq |U|$. If we are only interested in solutions for j up to some given m, then the algorithm can be implemented in $O(n \log m)$ time by keeping at most m nodes in the BST. We achieve this by deleting the rightmost node when the number of nodes exceeds m. Note that if the jobs are not already sorted by slopes then we still need to spend $O(n \log n)$ time.

If we would like to find out the weights of the MWMCMs of $G_{|U|,j}$ for all j such that $1 \leq j \leq |U|$, a naive approach would be to sort all prefixes of $\sigma_{|U|}$ and to compute the weights. Section 4.2.4 explains how to compute all those weights incrementally in linear time.

4.2.4 Incrementally Computing the Weights for All Prefixes of Slots

Let n denote $|U|$. Once we have computed σ_n, we know a set of jobs that forms an MWMCM of each $G_{n,j}$ for $1 \leq j \leq n$, but we are not readily given the weight $W(n, j)$ of such a matching. In this section, we describe how to compute the weights $W(n, 1), \ldots, W(n, n)$ incrementally in linear time by scanning through σ_n. It is straightforward to compute $W(n, 1)$. When we inspect the job at position j in σ_n (which is $\sigma_n[j]$), we can compute $W(n, j)$ in constant time from $W(n, j-1)$ because Lemma 4.2.5 implies that we can construct an MWMCM for $G_{n,j}$, namely $\mathit{matching}(\mathit{best}(n, j))$, from $\mathit{matching}(\mathit{best}(n, j-1))$, which is an MWMCM for $G_{n,j-1}$, by introducing $\sigma_n[j]$. Let u' denote $\sigma_n[j]$ and let i' denote the index of $\sigma_n[j]$. The change in weight caused by introducing job u' to $\mathit{matching}(\mathit{best}(n, j-1))$ in order to construct $\mathit{matching}(\mathit{best}(n, j))$ has two components: (1) $w(u', v')$ where v' denotes the slot with index $|\mathit{better}(u')|+1$, since Lemma 4.2.6 implies that job u' is matched to the slot with index $|\mathit{better}(u')| + 1$ in $\mathit{matching}(\mathit{best}(n, j))$; (2) the sum of the slopes of all jobs in $\mathit{best}(n, j)$ with indices greater than that of u', since each such job is matched to a slot with index one higher in $\mathit{matching}(\mathit{best}(n, j))$ than in $\mathit{matching}(\mathit{best}(n, j-1))$ and every other job is matched to the same slot in both matchings. By Lemma 4.2.6, the latter sum is equal to $\mathit{sum}(\mathit{best}(n, j-1)) - \mathit{sum}(\mathit{better}(i'))$. Since we scan the jobs in σ_n starting from position 1, we can maintain the sum of the slopes of the jobs scanned so far, thus we know $\mathit{sum}(\mathit{best}(n, j-1))$ when we reach job u'. We already store $|better(u')|$ and $sum(better(i'))$ at the node representing u', thus the change in weight caused by introducing the job u' can be computed in constant time.

4.3 Introducing Tardiness Penalties

Consider the following extension to the complete bipartite graphs $G = (U, V, w)$ introduced in Section 4.2. For each slot (right vertex) v in V, we introduce an integer parameter $v.quality$. We assume that the slots are indexed from 1 in non-decreasing order of qualities, breaking ties arbitrarily. We also assume that the qualities of the slots form a non-decreasing sequence that is the concatenation of two arithmetic sequences. We allow an arbitrary number of slots that is less than the number of jobs. We modify the edge weights so that $w(u, v)$ between job u and slot v becomes $u.intercept + u.slope \cdot v.quality$. We are able to solve the MWM problem in such a graph instance in $O(n \log n)$ time, which enables us to solve Problem 1.2 in $O(n \log n)$ time. The key idea is to utilize the incremental computation of the acceptance orders so that we can find the weights of the MWMCMs between the slots whose qualities form the first arithmetic sequence (the slots before the common due date) and every possible prefix of jobs. Then we do the same between the slots whose qualities form the second arithmetic sequence (the slots after the common due date) and every possible suffix of jobs. Then in linear time we find an optimal matching by determining which jobs to assign to the first group of slots and which jobs to the second group.

We start by introducing two families of graphs, $\mathcal{H}$ and $\mathcal{H}^*$: $\mathcal{H}$ is equivalent to UDALEWs (modulo the order of vertices, and the bid and item *id*'s) and it extends the family $\mathcal{G}$ by introducing qualities to the slots; $\mathcal{H}^*$ is the family of graphs mentioned in the previous paragraph and it consists of the graphs on which we encode the instances of Problem 1.2 as weighted matching problems. Then we discuss an algorithm that solves the MWMCM problem on graphs drawn from the family $\mathcal{H}^*$.

We define $\mathcal{H}$ as the family of all complete edge-weighted bipartite graphs $G = (U, V, w)$ such that the following conditions hold: $|U| \geq |V|$; each job u in U has two associated integers $u.intercept$ and $u.slope$; the jobs are indexed from 1 in non-decreasing order of slopes, breaking ties arbitrarily; each slot v in V has an associated integer $v.quality$; the slots are indexed from 1 in non-decreasing order of qualities, breaking ties arbitrarily; the weight $w(u, v)$ of the edge between a job u and a slot v is equal to $u.intercept + u.slope \cdot v.quality$. Note that a graph $G = (U, V, w)$ in $\mathcal{H}$ admits an $O(|U|)$-space representation. Also note that an input graph $G = (U, V, w)$ to the algorithm presented in Section 4.2 can be interpreted as a graph belonging to the family $\mathcal{H}$ that has $|U|$ slots with qualities forming the arithmetic sequence $\langle 1, \ldots, |U| \rangle$. Observe that the same algorithm can also be used to find an MWMCM for the case in which the qualities form a different arithmetic sequence by scaling the slopes, setting qualities to the arithmetic sequence $\langle 1, \ldots, |U| \rangle$, and modifying the intercepts.

We now introduce the notion of a "splitting point", a key technical concept that underlies our algorithm. Let $G = (U, V, w)$ be a graph in $\mathcal{H}$. We define U_i, V_j, and $G_{i,j}$ in the same manner as we did for a graph in $\mathcal{G}$ in Section 4.2. For any integer i such that $1 \le i < |U|$, we define U_{-i} as the set $U \setminus U_i$. Similarly for any integer j such that $1 \le j < |V|$, we define V_{-j} as the set $V \setminus V_j$. For any integers i and j such that $1 \le j \le i < |U|$ and $j < |V|$, we define $G_{-i,-j}$ as the subgraph of G induced by the vertices $U_{-i} \cup V_{-j}$. Then it is not hard to see that for any j in the range $1 \le j < |V|$, there exists at least one integer i, which we call a *splitting point* for j, such that the union $M_1 \cup M_2$ is an MWMCM of G where M_1 is any MWMCM of $G_{i,j}$ and M_2 is any MWMCM of $G_{-i,-j}$. Note that if i is a splitting point for j, then $|U_i| \ge |V_j|$ and $|U_{-i}| \ge |V_{-j}|$.

We encode an instance of Problem 1.2 as a weighted matching problem on a graph drawn from a family $\mathcal{H}^*$ that is contained in $\mathcal{H}$. We define $\mathcal{H}^*$ as the family of all graphs $G = (U, V, w)$ in $\mathcal{H}$ such that the qualities of the slots in V, when visited in index order, form a non-decreasing sequence that is the concatenation of two arithmetic sequences.

Let I be an instance of Problem 1.2. The instance I consists of a set of n jobs to schedule, each with a profit and a weight; a common due date d and a common deadline $\overline{d}$ where we assume that $d < \overline{d} \le n$; and a positive constant c. We encode the instance I as a graph $G = (U, V, w)$ in $\mathcal{H}^*$ such that the following conditions hold: $|U| = n$; $|V| = \overline{d}$; each u in U represents a distinct job in I; each v in V represents a distinct time slot in which a job in I can be scheduled; for each job in I and the vertex u that represents that job, $u.\mathit{intercept}$ is equal to the profit of the job and $u.\mathit{slope}$ is equal to the negated weight of the job; the qualities of the slots in V are set to form the concatenation of the arithmetic sequences $\langle 1, 2, \ldots, d\rangle$ and $\langle d+1+c, d+2+2c, \ldots, \overline{d}+(\overline{d}-d)c\rangle$. It is easy to see by inspecting the objective of Problem 1.2 that the instance I of Problem 1.2 is equivalent to the problem of finding an MWM of a graph $G = (U, V, w)$ in $\mathcal{H}^*$ that encodes I. Analogous to the case for $\mathcal{G}$ discussed in Section 4.2, the problem of finding an MWM of a graph in $\mathcal{H}^*$ can be reduced to the MWMCM problem by adding dummy jobs.

We now describe our algorithm for computing an MWMCM of a graph $G = (U, V, w)$ in $\mathcal{H}^*$. Let j denote the index such that the qualities of both V_j and V_{-j} are arithmetic sequences. If we can find the weights of MWMCMs of each $G_{i,j}$ for $j \leq i \leq |U|$ and the weights of MWMCMs of each $G_{-i,-j}$ for $1 \leq i \leq |U| - |V| + j$ in $O(|U| \log |U|)$ total time, then it takes linear time to find a splitting point for j, and thus an MWMCM of G can be constructed in $O(|U| \log |U|)$ total time.

Our algorithm consists of two extensions to the algorithm introduced for Problem 1.1. Let $G' = (U, V', w)$ be a graph in $\mathcal{H}$ such that the qualities of V' form an arithmetic sequence. In the remainder of this section, we use the shorthand G'_i (resp., G'_{-i}) to denote the subgraph of G' induced by the vertices $U_i \cup V'$ (resp., $U_{-i} \cup V'$) for any integer i such that $1 \leq i \leq |U|$. The first extension, which we discuss in the next paragraph, exploits the incremental computation of the acceptance orders performed by the algorithm introduced for Problem 1.1, in order to compute the weights of MWMCMs of each G'_i for $|V'| \leq i \leq |U|$ in $O(|U| \log |U|)$ total time. The second extension, which we discuss in the final paragraph, finds the weights of MWMCMs of each G'_{-i} for $1 \leq i \leq |U| - |V'|$ in $O(|U| \log |U|)$ total time by a simple reduction so that the first extension is utilized. Then, by setting G' to the subgraph of G induced by the vertices $U \cup V_j$ (resp., $U \cup V_{-j}$) as an input to the first (resp., second) extension, these two extensions are used to find the weights of the MWMCMs of each $G_{i,j}$ for $j \leq i \leq |U|$ and the weights of MWMCMs of each $G_{-i,-j}$ for $1 \leq i \leq |U| - |V| + j$ in $O(|U| \log |U|)$ total time in order to compute an MWMCM of a graph $G = (U, V, w)$ in $\mathcal{H}^*$.

First we show how to modify the algorithm introduced for Problem 1.1 so that we can compute the weights of MWMCMs of each subgraph G'_i for $|V'| \leq i \leq |U|$ given a graph $G' = (U, V', w)$ in $\mathcal{H}$ such that the qualities of V' form an arithmetic sequence. As a preprocessing step, we scale the slopes and modify the intercepts so that the instance G' is transformed such that the qualities form the arithmetic sequence $\langle 1, \ldots, |V'| \rangle$, as mentioned in the observation after the definition of $\mathcal{H}$. In what follows, let T denote the BST that the algorithm introduced for Problem 1.1 maintains. We modify the algorithm so that we keep at most $|V'|$ nodes in T by discarding the rightmost node when necessary, as mentioned in Section 4.2.3. Due to these deletions, $\mathit{represent}(T, i)$ no longer holds for $i > |V'|$. However, it is easy to argue that, for $i > |V'|$, the BST T in the modified algorithm contains the first $|V'|$ jobs in σ_i (i.e., $\mathit{best}(i, |V'|)$), and that these jobs occur in the same order (with respect to an inorder traversal) as in the BST in the unmodified algorithm. For any integer i such that $i \geq |V'|$, let M_i denote $\mathit{matching}(\mathit{best}(i, |V'|))$. Then Lemma 4.2.5 implies that M_i is an MWMCM of G'_i. We maintain an additional BST τ that concurrently stores the same set of jobs that are present in the main BST T, however in a different order. The keys of the nodes in τ are the indices of the corresponding jobs, thus an inorder traversal of τ yields an increasing order of indices. We implement τ as a balanced BST and augment it so that we can query for the sum of the slopes of all jobs that have indices greater than that of a given job. All the insert, delete, and query operations can be implemented in logarithmic time using standard augmentation techniques [11, Chapter 14]. We utilize those queries in order to maintain the weight of M_i at each iteration $i \geq |V'|$ in the following way. First, the weight of $M_{|V'|}$ can be computed at the end of iteration $|V'|$ via an inorder traversal of τ. Now suppose that at the end of some iteration i for $i > |V'|$, the set of jobs in T is changed by an update consisting of insertion of job u, which is the job with index i, to T (also to τ) and removal of some job u' from T (also from τ). Let τ_{i-1} denote the state of the BST τ before this update. Let j' be the index of the slot that is matched to u' in M_{i-1}. Note that j' is the rank of u' in τ_{i-1}. Let U' denote the set of jobs in τ_{i-1} with indices greater than that of u'. Since u has the highest index among the jobs that are matched in M_i, each job in U' is matched to a slot with index one lower in M_i than in M_{i-1}, and every other job that is matched in M_{i-1} except u' is matched to the same slot in M_i. Then, the weight of M_i minus the weight of

M_{i-1} is equal to $u.intercept + u.slope \cdot |V'| - u'.intercept - u'.slope \cdot j' - sum(U')$. Since such an update to τ and a query for $sum(U')$ in τ can be performed in $O(\log |V'|)$ time, we can maintain the weight of each M_i for $i \geq |V'|$ without slowing down the algorithm asymptotically.

In order to compute the weights of MWMCMs of each subgraph G'_{-i} for $1 \leq i \leq |U| - |V'|$ given a graph $G' = (U, V', w)$ in $\mathcal{H}$ such that the qualities of V' form an arithmetic sequence, we create another instance G'' by negating both the job slopes and slot qualities, and by reindexing both the jobs and the slots in reverse orders. Then we run the algorithm described in the previous paragraph on G''. Note that the weight of the edge between the job with an index i and the slot with an index j in G'' is equal to the weight of the edge between the job with index $|U| - i + 1$ and the slot with index $|V'| - j + 1$ in G'. Thus the weight of an MWMCM of G''_i is equal to the weight of an MWMCM of $G'_{-|U|+i}$ for $|V'| \leq i < |U|$.

4.4 NP-Hardness Results

It is natural to consider certain other problems within the setting of Problem 1.1, but with the goal of optimizing various other related criteria, possibly by imposing some constraints. Shabtay et al. [52] split the scheduling objective into two criteria: the scheduling cost f, which depends on the completion times of the jobs, and the rejection cost g, which is the sum of the penalties paid for the rejected jobs. In addition to the problem of minimizing $f + g$, Shabtay et al. also analyze the following two problems: minimization of f

subject to $g \leq R$, where R is a given upper bound on the rejection cost; minimization of g subject to $f \leq K$, where K is a given upper bound on the value of the scheduling criterion. In this section, we show that Problem 1.1 becomes NP-hard if we split our criteria in the same manner and aim for optimizing one while bounding the other.

Recall that the input to Problem 1.1 may be viewed as a graph $G = (U, V, w)$ in $\mathcal{G}$ where the weight $w(u, v)$ of an edge between a job u in U and a slot v in V with an index j is equal to $u.intercept + u.slope \cdot j$. We split the expression $w(u, v) = u.intercept + u.slope \cdot j$ denoting the weight of an edge (u, v) into two summands: the first term $u.intercept$, which we call the *profit component*; the second term $u.slope \cdot j$, which we call the *scheduling component*. For a given MCM M of a graph G in $\mathcal{G}$, we define $f(G, M)$ as the sum of the scheduling components of the weights of the edges in M, and we define $g(G, M)$ as the sum of the profit components of the weights of the edges in M.

Given a graph G in $\mathcal{G}$, let $\mathcal{M}_G$ denote the set of all MCMs of G. Then we define the following three problems, which are analogous to the problems mentioned above from [52].

- P1: Find a matching M in $\mathcal{M}_G$ maximizing $f(G, M) + g(G, M)$.
- P2: Find a matching M in $\mathcal{M}_G$ maximizing $f(G, M)$ subject to $g(G, M) \geq R$.
- P3: Find a matching M in $\mathcal{M}_G$ maximizing $g(G, M)$ subject to $f(G, M) \geq$

K.

The algorithm we introduced for Problem 1.1 solves P1 in $O(n \log n)$ time, where n denotes the number of jobs in G. In this section, we show that P2 and P3 are NP-hard. We define the decision version of both P2 and P3 as follows: Given a graph G in $\mathcal{G}$ and two integers K and R, is there an MCM M of G such that $f(G, M) \geq K$ and $g(G, M) \geq R$? In what follows, we refer to this decision problem as DP.

We show the NP-hardness of P2 and P3 by reducing the partition problem, which is known to be NP-complete, to DP. The partition problem is defined as follows: Given a sequence ρ of m positive integers $\langle \rho_1, \ldots, \rho_m \rangle$ with sum $\sum_{i=1}^{m} \rho_i = 2W$, is there a subsequence of ρ with sum W? We assume $m \geq 2$ and $\rho_i \leq W$ for all $1 \leq i \leq m$.

Throughout the remainder of the section, we fix an arbitrary instance ρ of this partition problem. We now describe how to transfer ρ to an instance (G, K, R) of DP. Our description introduces a variety of symbols, all of which are fixed in value, throughout the remainder of this section.

Let m denote the size of ρ. Let W denote $\frac{1}{2} \sum_{i=1}^{m} \rho_i$. For any integer i such that $1 \leq i \leq m$, let A_i denote $-2^{i-1}W$. Note that $A_i = \sum_{j=1}^{i-1} A_j - W$. For any integer i such that $1 \leq i \leq m$, let B_i denote $3^i W$.

Let G be a graph in $\mathcal{G}$ with a set U of $2m$ jobs $\{u_i, \ldots, u_{2m}\}$, and a set V of m slots $\{v_i, \ldots, v_m\}$, and where the intercepts and slopes are determined as follows. For any i such that $1 \leq i \leq m$, we define the intercept of job u_{2i-1} as $a_{2i-1} = A_i$, the intercept of job u_{2i} as $a_{2i} = A_i - \rho_i$, the slope of job u_{2i-1} as

$b_{2i-1} = B_i$, and the slope of job u_{2i} as $b_{2i} = B_i + \frac{\rho_i}{i}$. Thus, for a given MCM M of G,

$$f(G, M) = \sum_{(u_i, v_j) \in M} b_i \cdot j,$$

and

$$g(G, M) = \sum_{(u_i, v_j) \in M} a_i.$$

Let K denote $\sum_{i=1}^{m} iB_i + W$, and let R denote $\sum_{i=1}^{m} A_i - W = -2^m W$. It is straightforward to verify that (G, K, R) is a DP instance, and the transformation from ρ to (G, K, R) can be performed in polynomial time.

Lemma 4.4.1. If ρ is a positive instance of the partition problem, then (G, K, R) is a positive instance of DP.

Proof. Assume that ρ is a positive instance of the partition problem. Let S be a subsequence of ρ with sum W. We construct an MCM M of G as follows: For any i such that $1 \leq i \leq m$, if ρ_i is in S then match u_{2i} with v_i; otherwise, match u_{2i-1} with v_i. It is easy to verify that $f(G, M) = K$ and $g(G, M) = R$. □

Lemma 4.4.2. Let U' be a size-m subset of U. Then among all the MCMs of G matching the jobs U', there is a unique matching M that maximizes $f(G, M)$, and this unique M matches the jobs to the slots $v_1, \ldots, v_m$ in increasing order of indices.

Proof. Observe that $b_i > b_{i-1}$ for any i such that $1 < i \leq 2m$. Hence the slopes of the jobs in U' are distinct. Then the result follows from the rearrangement inequality [34, Section 10.2, Theorem 368]. □

The following technical lemma is used in the proof of Lemma 4.4.4.

Lemma 4.4.3. $\sum_{j=1}^{i} jb_{i+j-2} \le iB_i$.

Proof. For any j such that $1 \le j \le m-1$, we have $(j+1)b_{2j} + jb_{2j-1} \le 2(j+1)B_j$ since

$$\begin{aligned}(j+1)b_{2j} + jb_{2j-1} &= (j+1)\left(B_j + \frac{\rho_j}{j}\right) + jB_j \\ &\le (j+1)B_j + 2\rho_j + (j+1)B_j - B_j \\ &\le 2(j+1)B_j + 2W - 3W \\ &\le 2(j+1)B_j.\end{aligned}$$

Thus

$$\begin{aligned}\sum_{j=1}^{i} jb_{i+j-2} &\le \sum_{j=1}^{i-1}(j+1)b_{2j} + jb_{2j-1} \\ &\le 2\sum_{j=1}^{i-1}(j+1)B_j \\ &\le 2iB_{i-1}\sum_{j\ge 0} 3^{-j} \\ &= 3iB_{i-1} \\ &= iB_i.\end{aligned}$$

□

Lemma 4.4.4. Let M be an MCM of G such that $f(G, M) \geq K$ and $g(G, M) \geq R$. If (G, K, R) is a positive instance of DP, then for each i such that $1 \leq i \leq m$, exactly one of u_{2i-1} and u_{2i} is matched in M, and it is matched to v_i.

Proof. Assume that (G, K, R) is a positive instance of DP. Let $P_1(i)$ denote the predicate "at least one of the jobs u_{2i-1} and u_{2i} is matched in M", and let $P_2(i)$ denote the predicate "at most one of the jobs u_{2i-1} and u_{2i} is matched in M". Then we claim the following.

Claim 1: If $\bigwedge_{j=i+1}^{m} (P_1(j) \wedge P_2(j))$ holds for some integer i such that $1 \leq i \leq m$, then $P_1(i)$ holds. It is easy to prove the claim for $i = 1$. Let i be an integer such that $1 < i \leq m$ and assume that the claim does not hold. Then, $\bigwedge_{j=i+1}^{m} (P_1(j) \wedge P_2(j))$ holds and neither u_{2i-1} nor u_{2i} is matched in M. We derive an upper on $f(G, M)$ as follows. Let U' denote the set of jobs that are matched in M. We know that U' consists of exactly one job from each pair (u_{2j-1}, u_{2j}) for $i < j \leq m$, and i other jobs having indices less than $2i - 1$. Lemma 4.4.2 implies that the unique MCM M' that matches U' and that maximizes $f(G, M')$ has the following structure: for all $i < j \leq m$, the job from the pair (u_{2j-1}, u_{2j}) that is present in U' is matched to the slot v_j; the remaining i jobs in U' are assigned to the slots $v_1, \ldots, v_i$ in increasing order of indices. Let M^* denote this unique MCM, thus $f(G, M) \leq f(G, M^*)$. We construct another matching M'' by assigning u_{2j} to v_j for $i < j \leq m$, and by assigning $u_{i-1}, \ldots, u_{2i-2}$ to $v_1, \ldots, v_i$. An upper bound on $f(G, M'')$ is $\sum_{j=1}^{i} jb_{i+j-2} + \sum_{j=i+1}^{m} jB_j + 2W$, where the first term comes from the subset of M'' involving $v_1, \ldots, v_i$, and the rest is an upper bound for the subset of M''

involving $v_{i+1}, \ldots, v_m$. It is easy to see that $f(G, M^*) \le f(G, M'')$ since for each slot v, either both M'' and M^* match the same job to v, or the job that M'' matches to v has a slope greater than that of the job that M^* matches to v. Thus

$$\begin{aligned} f(G, M) &\le f(G, M'') \\ &\le \sum_{j=1}^{i} j b_{i+j-2} + \sum_{j=i+1}^{m} j B_j + 2W \\ &= \sum_{j=1}^{i} j b_{i+j-2} + K - \sum_{j=1}^{i} j B_j + W \\ &\le i B_i + K - \sum_{j=1}^{i} j B_j + W \\ &= K - \sum_{j=1}^{i-1} j B_j + W \\ &< K, \end{aligned}$$

where the fourth line follows from Lemma 4.4.3. This contradicts $f(G, M) \ge K$.

Claim 2: If $\bigwedge_{j=i+1}^{m} (P_1(j) \wedge P_2(j))$ holds for some integer i such that $1 \le i \le m$, then $P_2(i)$ holds. It is easy to prove the claim for $i = 1$. Let i be an integer such that $1 < i \le m$ and assume that the claim does not hold. Then, $\bigwedge_{j=i+1}^{m} (P_1(j) \wedge P_2(j))$ holds and both u_{2i-1} and u_{2i} are matched in M. Then, $g(G, M)$ is at most $2A_i - \rho_i + \sum_{j=i+1}^{m} A_j$. Using the equality $A_i = \sum_{j=1}^{i-1} A_j - W$, we deduce that $g(G, M)$ is at most $\sum_{j=1}^{m} A_j - W - \rho_i$, contradicting $g(G, M) \geq R$ since ρ_i is positive.

Claim 3: For each integer i such that $1 \leq i \leq m$, exactly one job from the pair (u_{2i-1}, u_{2i}) is matched in M. This claim is easily seen to hold by reverse induction on i using Claims 1 and 2.

Let U' denote the set of jobs that are matched in M. Lemma 4.4.2 and Claim 3 imply that the unique MCM M' that matches U' and that maximizes $f(G, M')$ matches exactly one job from each pair (u_{2i-1}, u_{2i}) to v_i for $1 \leq i \leq m$. Let M^* denote this unique MCM. It is easy to argue that the maximum $f(G, M')$ a matching M' that matches U' can attain is K. Since $f(G, M)$ is at least this maximum, M is M^*. $\square$

Lemma 4.4.5. If (G, K, R) is a positive instance of DP, then ρ is a positive instance of the partition problem.

Proof. Assume that (G, K, R) is a positive instance of DP. Let M be an MCM of G such that $f(G, M) \geq K$ and $g(G, M) \geq R$. We construct a subsequence S of ρ as follows. We iterate over the slots in G from lowest index to the highest. Lemma 4.4.4 implies that a slot v_i is matched either to u_{2i-1} or to u_{2i} in M. We include ρ_i in the subsequence S if and only if v_i is matched to u_{2i} in M. Let $\sum_S$ denote the sum of the integers in the subsequence S. Then it is easy to verify that $f(G, M) = \sum_{i=1}^{m} iB_i + \sum_S$ and $g(G, M) = \sum_{i=1}^{m} A_i - \sum_S$. Finally, $f(G, M) \geq K$ implies that $\sum_S \geq W$, and $g(G, M) \geq R$ implies that $\sum_S \leq W$. Hence $\sum_S = W$. $\square$

Theorem 4.4.6. The optimization problems P2 and P3 are NP-hard.

Proof. Immediate from Lemmas 4.4.1 and 4.4.5, since DP is the decision version of both P2 and P3. □

Part II

Unit-Demand Auctions and Stable Marriage with Indifferences

Chapter 5

Strategyproof Pareto-Stable Mechanisms for Two-Sided Matching with Indifferences

In the remainder of this dissertation, we explore a connection between unit-demand auctions and the stable marriage model (and more generally, the college admissions model). This chapter presents the first mechanisms that enjoy a strong combination of game-theoretic properties, namely strategyproofness and Pareto-stability, for the stable marriage model with incomplete and weak preferences (allowing indifferences in the agents' preferences) and the college admissions model. An abbreviated version of the results presented in this chapter appears in a workshop publication [18].

In Section 5.1, we briefly review some related work. In Sections 5.2

and 5.3, we introduce a framework based on two variants of unit-demand auctions to generalize the deferred acceptance algorithm to allow indifferences: Section 5.2 defines the notion of a unit-demand auction with priorities (UAP); Section 5.3 builds on the UAP notion to define the notion of an iterated UAP (IUAP), and establishes a number of important properties of IUAPs. Building on this framework, Section 5.4 presents our polynomial-time algorithm for the stable marriage problem with indifferences that provides a strategyproof Pareto-stable mechanism. Section 5.5 presents our polynomial-time algorithm for the college admissions problem that provides a strategyproof Pareto-stable mechanism assuming that the preferences of the colleges are minimally responsive.

5.1 Related Work

Erdil and Ergin [26] and Chen and Ghosh [9] present polynomial-time algorithms for computing Pareto-stable matchings in certain two-sided matching models discussed shortly. These algorithms are based on a two-phase approach that was previously proposed by Sotomayor [61]. The first phase runs the DA algorithm after breaking the ties arbitrarily to obtain a weakly stable matching. The second phase repeatedly updates the matching via a sequence of Pareto improvements until no such improvement is possible. However, it is known that this two-phase framework does not yield a strategyproof mechanism [18].

Erdil and Ergin [26] consider the special case of the CAW model where the following restrictions hold for all students i and colleges j: i is not indifferent between being assigned to j and being left unassigned; j is not indifferent between having one of its slots assigned to i and having that slot left unfilled. We remark that this special case of CAW corresponds to the HRT problem discussed in Manlove [42, Chapter 3], which is stated using resident-hospital terminology. For this special case, Erdil and Ergin present a polynomial-time algorithm for computing a Pareto-stable matching when the preferences of the colleges satisfy a technical restriction related to responsiveness; the notion of responsiveness is introduced by Roth [49]. We consider the same class of preferences, which we refer to as *minimally responsive*; see Section 5.5 for a formal definition. The algorithm of Erdil and Ergin does not provide a strategyproof mechanism. Chen and Ghosh [9] build on the results of Erdil and Ergin by considering the many-to-many generalization of HRT in which the agents on both sides of the market have capacities, and the agent preferences are minimally responsive, though they do not use this terminology. For this generalization, Chen and Ghosh provide a *strongly* polynomial-time algorithm. No strategyproof mechanism (even for the agents on one side of the market) is possible in the many-to-many setting, since it is a generalization of the college admissions model with strict preferences. As in the work of Erdil and Ergin [26] and Chen and Ghosh [9], we assume that the preferences of the colleges are minimally responsive. We can also handle the class of college preferences "induced by additive utility" that is defined in Section 5.5.2.

In the many-to-many matching setting addressed by Chen and Ghosh [9], a pair of agents (on opposite sides of the market) can be matched with arbitrary multiplicity, as long as the capacity constraints are respected. Chen [8] presents a polynomial-time algorithm for the variation of many-to-many matching in which a pair of agents can only be matched with multiplicity one. Kamiyama [35] addresses the same problem using a different algorithmic approach based on rank-maximal matchings. The algorithms of Chen and Kamiyama are strongly polynomial, since we can assume without loss of generality that the capacity of any agent is at most the number of agents on the other side of the market. Since this variation of the many-to-many setting also generalizes the college admissions model with strict preferences, it does not admit a strategyproof mechanism, even for the agents on one side of the market.

Some real-world applications of matching models that are similar to the college admissions model include school choice systems in the United States and schemes that match medical residents to hospitals, such as the National Resident Matching Program (NRMP) [50] in the United States and the Canadian Resident Matching Service [1]. The evolving structure of the medical labor market has made the resident/hospital matching problem more complex over time. For example, in the NRMP market, couples seek positions in close proximity. See [50] for further discussion of practical considerations related to resident/hospital matching. Likewise, practical systems of school choice tend to take into account additional constraints. To give the reader a better sense of the interplay between theory and practice in systems of two-sided matching, below we discuss some recent work in the realm of the school choice.

Many public school districts in the United States, including New York City, Boston, Cambridge, Charlotte-Mecklenburg, Chicago, Columbus, Denver, Miami-Dade, Minneapolis, New York City, New Orleans, Newark, San Francisco, Seattle, and St. Petersburg-Tampa, have implemented centralized school choice systems with the goal of offering each student an equitable opportunity to attend their preferred schools. In many of these systems, the preferences of the students are strict, but the schools express rankings of students that include ties. An important difference between school choice systems and college admissions model is that, with a few exceptions like some schools in New York City, the rankings of the students by the schools are determined by local laws and education policies. Moreover, only the preferences of students are considered in the welfare criteria in school choice, since generally schools are considered as objects to be consumed by students. Motivated by this one-sided notion of welfare in school choice, Erdil and Ergin [25, 26] and Kesten [37] consider a second natural solution concept in addition to Pareto-stability. In the context of stable marriage with indifferences, this solution concept seeks a weakly stable matching M that is "man-optimal" in the following sense: for all weakly stable matchings M', either all of the men are indifferent between M and M', or at least one man prefers M to M'. Erdil and Ergin [26] present a polynomial-time algorithm to compute such a man-optimal weakly stable matching for CAW. Erdil and Ergin [25] and Kesten [37] prove that no strategyproof man-optimal weakly stable mechanism exists for SMCW.

Many school choice systems employ the student-proposing DA algorithm where ties in the preferences of the schools are broken using random priorities assigned to the students. Abdulkadiroğlu et al. [2] study the New York City high school match data from school years 2003-04 to 2006-07, and conclude that the manner in which ties are broken has a significant impact on the quality of the matching from the perspective of the students. The quality of the matching can also be evaluated from the perspective of the social planner. Some complaints about the school choice systems, e.g., in Boston, include high transportation costs, "illusion of choice" (families are presented with a large set of schools to rank, even though many of these schools may be unattainable in practice), and low community cohesion (all students living on the same street might go to different schools, so local community is weakened) [56]. Pathak and Shi [43] mathematically model the key trade-offs in the 2012-2013 Boston school choice reform as an optimization problem, and build a discrete choice model to predict how families rank schools. Using this prediction model, the authors forecast the performance of various proposed plans by simulation, evaluating these plans in terms of equity of access to quality, proximity to home, variety of choice, predictability, bus coverage area, socio-economic diversity, and community cohesion. The authors propose a "Home-Based" plan to determine the set of schools that each family is asked to rank. A variant of this plan was adopted in Boston in 2014. Ashlagi and Shi [6] study the allocation of heterogeneous services to agents without monetary transfers, where the goal of the social planner is to maximize a potentially complex public objective. For tractability, the authors adopt an engineering approach by first solving a large-market approximation, and then converting the solution to a finite-market mechanism. The authors apply their framework to real data from Boston to design a mechanism that assigns students to public schools and maximizes a linear combination of utilitarian and max-min welfare, subject to capacity and transportation constraints. They report improvements in utilitarian welfare by an amount equivalent to decreasing students average distance to school by 0.5 miles, and in max-min welfare by about 2.5 miles. Ashlagi and Shi [5] study school choice from the perspective of improving community cohesion. The random priorities assigned to the students to break ties in the preferences of the schools induce a probability of each student being assigned to each school. The authors propose to improve community cohesion by implementing a "correlated lottery". They show how to find a convex combination of deterministic assignments that improves community cohesion, i.e., increases the number of pairs of students from the same community going to the same school, while maintaining the original assignment probabilities. The authors show that maximizing the community cohesion while maintaining the assignment probabilities is NP-hard even with two schools, and they present a heuristic that performs well in practice. Using simulations based on 2012 Boston school choice data, they report substantial increase in cohesion for new families for kindergarten 1 and kindergarten 2 (79% for K1 and 37% for K2).

5.2 Unit-Demand Auctions with Priorities

In this section, we formally define the notion of a unit-demand auction with priorities (UAP). In Section 5.2.1, we describe an associated matroid for a given UAP and we use this matroid to define the notion of a "greedy MWM". In Section 5.2.2, we establish a result related to extending a given UAP by introducing additional bidders. In Section 5.2.3, we discuss how to efficiently compute a greedy MWM in a UAP. In Section 5.2.4, we introduce a key definition that is helpful for establishing our strategyproofness results. We start with some useful definitions.

A *(unit-demand) bid β for a set of items V* is a subset of $V \times \mathbb{R}$ such that no two pairs in β share the same first component. (So β may be viewed as a partial function from V to $\mathbb{R}$.)

A *bidder u for a set of items V* is a triple (α, β, z) where α is an integer ID, β is a bid for V, and z is a real priority. For any bidder $u = (\alpha, \beta, z)$, we define $\mathit{id}(u)$ as α, $\mathit{bid}(u)$ as β, $\mathit{priority}(u)$ as z, and $\mathit{items}(u)$ as the union, over all (v, x) in β, of $\{v\}$.

A *unit-demand auction with priorities (UAP)* is a pair $A = (U, V)$ satisfying the following conditions: V is a set of items; U is a set of bidders for V; each bidder in U has a distinct ID.

5.2.1 An Associated Matroid

A UAP $A = (U, V)$ may be viewed as an edge-weighted bipartite graph, where the set of edges incident on bidder u correspond to $\mathit{bid}(u)$: for each pair

(v, x) in $bid(u)$, there is an edge (u, v) of weight x. We refer to a matching (resp., maximum-weight matching (MWM), maximum-cardinality MWM (MCMWM)) in the associated edge-weighted bipartite graph as a matching (resp., MWM, MCMWM) of A. For any edge $e = (u, v)$ in a given UAP, the associated weight is denoted $w(e)$ or $w(u, v)$. For any set of edges E, we define $w(E)$ as $\sum_{e \in E} w(e)$. For any UAP A, we let $w(A)$ denote the weight of an MWM of A.

Lemma 5.2.1. Let $A = (U, V)$ be a UAP, and let $\mathcal{I}$ denote the set of all subsets U' of U such that there exists an MWM of A that matches every bidder in U'. Then $(U, \mathcal{I})$ is a matroid.

Proof. The only nontrivial property to show is the exchange property. Assume that U_1 and U_2 belong to $\mathcal{I}$ and that $|U_1| > |U_2|$. Let M_1 be an MWM of A that matches every bidder in U_1, and let M_2 be an MWM of A that matches every bidder in U_2. If M_2 matches some bidder u in $U_1 \setminus U_2$, then $U_2 + u$ belongs to $\mathcal{I}$, as required. Thus, in what follows, we assume that M_2 does not match any bidder in $U_1 \setminus U_2$. The symmetric difference of M_1 and M_2, denoted $M_1 \oplus M_2$, corresponds to a collection of vertex-disjoint paths and cycles. Since M_2 does not match any bidder in $U_1 \setminus U_2$, we deduce that each bidder in $U_1 \setminus U_2$ is an endpoint of one of the paths in this collection. Since $|U_1| > |U_2|$, $|U_1 \setminus U_2| = |U_1| - |U_1 \cap U_2|$, and $|U_2 \setminus U_1| = |U_2| - |U_1 \cap U_2|$, we have $|U_1 \setminus U_2| > |U_2 \setminus U_1|$. It follows that there is at least one path in this collection, call it P, such that one endpoint of P is a bidder u in $U_1 \setminus U_2$ and the other endpoint of P is a vertex y that does not belong to $U_2 \setminus U_1$.

Moreover, y does not belong to U_1: if the length of P is odd, then y is an item and hence does not belong to U_1; if the length of P is even, then y is not matched in M_1 and hence does not belong to U_1. Since y does not belong to $U_2 \setminus U_1$ and does not belong to U_1, we conclude that y does not belong to U_2. The edges of P alternate between M_1 and M_2. Let X_1 denote the edges of P that belong to M_1, and let X_2 denote the edges of P that belong to M_2. Since M_1 is an MWM of A and $M'_1 = M_1 \oplus P = (M_1 \cup X_2) \setminus X_1$ is a matching of A, we deduce that $w(X_1) \geq w(X_2)$. Since M_2 is an MWM of A and $M'_2 = M_2 \oplus P = (M_2 \cup X_1) \setminus X_2$ is a matching of A, we deduce that $w(X_2) \geq w(X_1)$. Hence $w(X_1) = w(X_2)$ and M'_1 and M'_2 are MWMs of A. The MWM M'_2 matches all of the vertices on P except for y. Since y does not belong to U_2, we conclude that M'_2 matches all of the vertices in $U_2 + u$, and so the exchange property holds. $\square$

For any UAP A, we define *matroid*(A) as the matroid of Lemma 5.2.1.

For any UAP $A = (U, V)$ and any independent set U' of *matroid*(A), we define the *priority of* U' as the sum, over all bidders u in U', of *priority*(u). For any UAP A, the matroid greedy algorithm can be used to compute a maximum-priority maximal independent set of *matroid*(A).

For any matching M of a UAP $A = (U, V)$, we define *matched*(M) as the set of all bidders in U that are matched in M. We say that an MWM M of a UAP A is *greedy* if *matched*(M) is a maximum-priority maximal independent set of *matroid*(A). For any UAP A, we define the predicate *unique*(A) to hold if *matched*$(M) =$ *matched*(M') for all greedy MWMs M and M' of A.

For any matching M of a UAP, we define the *priority of* M, denoted *priority*(M), as the sum, over all bidders u in *matched*(M), of *priority*(u). Thus an MWM is greedy if and only if it is a maximum-priority MCMWM.

Lemma 5.2.2. All greedy MWMs of a given UAP have the same distribution of priorities.

Proof. This is a standard matroid result that follows easily from the exchange property and the correctness of the matroid greedy algorithm. $\square$

For any UAP A and any real priority z, we define *greedy*(A, z) as the (uniquely defined, by Lemma 5.2.2) number of matched bidders with priority z in any greedy MWM of A.

Lemma 5.2.3. Let $A = (U, V)$ be a UAP. Let u be a bidder in U such that (v, x) belongs to *bid*(u), *priority*$(u) = z$, and u is not matched in any greedy MWM of A. Let u' be a bidder in U such that (v, x') belongs to *bid*(u'), *priority*$(u') = z'$, and u' is matched to v in some greedy MWM of A. Then $(x, z) < (x', z')$.[1]

Proof. Let M be a greedy MWM in which u' is matched to v. Thus u is not matched in M. Let M' denote the matching $M - (u', v) + (u, v)$. Since M is an MCMWM of A and $w(M') = w(M) - x' + x$, we conclude that $x \leq x'$. If $x < x'$, the claim of the lemma follows. Assume that $x = x'$. In this case, M' is an MCMWM of A since $w(M') = w(M)$ and $|M'| = |M|$. Since M is a greedy MWM of A and *priority*(M') = *priority*$(M) - z' + z$, we conclude

[1]Throughout this chapter, comparisons of pairs are to be performed lexicographically.

that $z \leq z'$. If $z = z'$ then M' is a greedy MWM of A that matches u, a contradiction. Hence $z < z'$, as required. □

5.2.2 Extending a UAP

Let $A = (U, V)$ be a UAP and let u be a bidder such that $id(u)$ is not equal to the ID of any bidder in U. Then we define $A + u$ as the UAP $(U + u, V)$. For any UAPs $A = (U, V)$ and $A' = (U', V')$, we say that A' *extends* A if $U \subseteq U'$ and $V = V'$.

Lemma 5.2.4. Let $A = (U, V)$ be a UAP, let u be a bidder in U that is not matched in any greedy MWM of A, and let $A' = (U', V)$ be a UAP that extends A. Then u is not matched in any greedy MWM of A'.

Proof. Suppose u is matched in a greedy MWM, call it M_1, of A'. In what follows, we derive a contradiction by proving that u is matched in some greedy MWM of A. Let M_0 denote a greedy MWM of A. If u is matched in M_0, we are done, so assume that u is not matched in M_0. Thus $M_0 \oplus M_1$ contains a unique path P with u as an endpoint. The edges of P alternate between M_0 and M_1. Let X_0 denote the edges of P that belong to M_0, and let X_1 denote the edges of P that belong to M_1.

Since u is matched in M_1 and not in M_0, the other endpoint of P is either an item, or it is a bidder that is matched in M_0 and not in M_1. Either way, we deduce that all of the vertices on P belong to A. Thus $M_0' = M_0 \oplus P = (M_0 \cup X_1) \setminus X_0$ is a matching in A. Since M_0 is an MWM of A and M_0' is a matching of A, we deduce that $w(M_0) \geq w(M_0')$ and hence that

$w(X_0) \geq w(X_1)$. Since all of the vertices on P belong to A', we conclude that $M_1' = M_1 \oplus P = (M_1 \cup X_0) \setminus X_1$ is a matching in A'. Since M_1 is an MWM of A' and M_1' is a matching of A', we deduce that $w(M_1) \geq w(M_1')$ and hence that $w(X_1) \geq w(X_0)$. Thus $w(X_0) = w(X_1)$, and we conclude that M_0' is an MWM of A.

Since u is matched in M_1 and not in M_0, we deduce that $|X_1| \geq |X_0|$ and hence that $|M_0'| \geq |M_0|$. Since M_0 is a greedy MWM of A, we know that M_0 is an MCMWM of A, and hence that $|M_0| \geq |M_0'|$. Thus $|M_0| = |M_0'|$ and hence $|X_0| = |X_1|$ and M_0' is an MCMWM of A. Since $|X_0| = |X_1|$, the other endpoint of P is a bidder u' that is matched in M_0 and not in M_1. Since M_0 is a greedy MWM of A and M_0' is an MCMWM of A, we deduce that $\mathit{priority}(M_0) \geq \mathit{priority}(M_0')$ and hence that $\mathit{priority}(u') \geq \mathit{priority}(u)$.

Since $|X_0| = |X_1|$ and $w(X_0) = w(X_1)$, we deduce that M_1' is an MCMWM of A'. Since M_1 is a greedy MWM of A' and M_1' is an MCMWM of A', we deduce that $\mathit{priority}(M_1) \geq \mathit{priority}(M_1')$ and hence that $\mathit{priority}(u) \geq \mathit{priority}(u')$. Since we argued above that $\mathit{priority}(u') \geq \mathit{priority}(u)$, we conclude that $\mathit{priority}(u) = \mathit{priority}(u')$, and hence that M_0' is a greedy MWM of A. This completes the proof, since u is matched in M_0'. □

5.2.3 Finding a Greedy MWM

In this section, we briefly discuss how to efficiently compute a greedy MWM of a UAP via a slight modification of the classic Hungarian method for the assignment problem [39]. In the (maximization version of the) assignment problem, we are given a set of n agents, a set of n tasks, and a weight for each agent-task pair, and our objective is to find a perfect matching (i.e., every agent and task is required to be matched) of maximum total weight. The Hungarian method for the assignment problem proceeds as follows: a set of dual variables, namely a "price" for each task, and a possibly incomplete matching are maintained; an arbitrary unmatched agent u is chosen and a shortest augmenting path from u to an unmatched task is computed using "residual costs" as the edge weights; an augmentation is performed along the path to update the matching, and the dual variables are adjusted in order to maintain complementary slackness; the process repeats until a perfect matching is found.

Within our UAP setting, the set of bidders can be larger than the set of items, and some bidder-item pairs may not be matchable, i.e., the associated bipartite graph is not necessarily complete. In this setting, we can use an "incremental" version of the Hungarian method to find an (not necessarily greedy) MWM of a given UAP $A = (U, V)$ as follows. For the purpose of simplifying the presentation of our method, we enlarge the set of items by adding a dummy item v_0 such that v_0 is connected to each bidder u with an edge of weight $w(u, v_0) = 0$ and we always maintain v_0 in the residual graph with a price of 0. We start with the empty matching M. Then, for each bidder u in U (in arbitrary order), we process u via an "incremental Hungarian step" as follows: let U' denote the set of bidders that are matched by M; let V' denote the set of items that are not matched by M; find the shortest paths from u to each item v in $V' + v_0$ in the residual graph; let W denote the minimum path weight among these shortest paths; choose a path P that is either (1) a shortest path of weight W from u to an item v in V', or (2) a shortest path from u to a bidder u' in $U' + u$ such that extending P with the edge (u', v_0) yields a shortest path of weight W from u to v_0; augment M along P; adjust the prices in order to maintain complementary slackness; update the residual graph. The algorithm terminates when every non-reserve bidder has been processed. The algorithm performs $|U|$ incremental Hungarian steps and each incremental Hungarian step can be implemented in $O(|V| \log |V| + m)$ time by utilizing Fibonacci heaps [28], where m denotes the number of edges in the residual graph, which is $O(|V|^2)$.

In order to find a greedy MWM, we slightly modify the implementation described in the previous paragraph. Lemmas 5.2.7 and 5.2.8 established below imply that choosing the path P in the following way results in a greedy MWM: if a path of type (1) exists, we arbitrarily choose such a path; if no path of type (1) exists, then we identify the nonempty set U'' of all bidders u' such that a path of type (2) exists, and we choose a shortest path P that terminates at a minimum priority bidder in U''. It is easy to see that the described modification does not increase the asymptotic time complexity of the algorithm. In the remainder of this section, we establish Lemmas 5.2.7, 5.2.8, and 5.2.9; Lemma 5.2.9 is used in Section 5.3.3 to prove Lemma 5.3.9. We start with some useful definitions.

Let $A = (U, V)$ and $A' = A + u$ be UAPs, and let M be an MWM of A. We define *digraph*(A, u, M) as the edge-weighted digraph that may be obtained by modifying the subgraph of A induced by the set of vertices $(\mathit{matched}(M) + u) \cup V$ as follows: for each edge that belongs to M, we direct the edge from item to bidder and leave the weight unchanged; for each edge that does not belong to M, we direct the edge from bidder to item and negate the weight.

Lemma 5.2.5. Let $A = (U, V)$ and $A' = A + u$ be UAPs, let M be an MWM of A, and let G denote $\mathit{digraph}(A, u, M)$. Then G does not contain any negative-weight cycles.

Proof. Such a cycle could not involve u (since u only has outgoing edges) so it has to be a negative-weight cycle that already existed before u was added, a contradiction since M is an MWM of A. □

Let $A = (U, V)$ and $A' = A + u$ be UAPs, let M be an MWM of A, and let G denote $\mathit{digraph}(A, u, M)$. We define a set of items $\mathit{holes}(A, u, M)$, and a set of bidders $\mathit{candidates}(A, u, M)$, as follows. By Lemma 5.2.5, the shortest path distance in G from bidder u to any vertex reachable from u is well-defined. We define $\mathit{holes}(A, u, M)$ as the set of all items v in V such that v is unmatched in M and the weight of a shortest path in G from u to v is $w(A) - w(A')$. We define $\mathit{candidates}(A, u, M)$ as the set of all bidders u' such that the weight of a shortest path in G from u to u' is equal to $w(A) - w(A')$.

Let $A = (U, V)$ and $A' = A + u$ be UAPs, let M be an MWM of A, and let P be a directed path in $\mathit{digraph}(A, u, M)$ that starts at u, has weight $w(A) - w(A')$, and terminates at either an item in $\mathit{holes}(A, u, M)$ or a bidder in $\mathit{candidates}(A, u, M)$. (Note that P could be a path of length zero from u to

u.) Let X denote the edges in M that correspond to item-to-bidder edges in P, and let Y denote the edges in A' that correspond to bidder-to-item edges in P. It is easy to see that the set of edges $(M \setminus X) \cup Y$ is an MWM of A'. We define this MWM of A' as *augment*(A, u, M, P).

Lemma 5.2.6. Let $A = (U, V)$ be a UAP, let M be a greedy MWM of A, let u be a bidder that does not belong to U, let A' denote the UAP $(U+u, V)$, and let M' denote a greedy MWM of A' minimizing $|M \oplus M'|$. Then *digraph*(A, u, M) contains a directed path P satisfying the following conditions: P has weight $w(A) - w(A')$; P starts at u; the bidder-to-item edges in P correspond to the edges in $M' \setminus M$; the item-to-bidder edges in P correspond to the edges in $M \setminus M'$; if *holes*(A, u, M) is nonempty, then P terminates at an item in *holes*(A, u, M); if *holes*(A, u, M) is empty, then P terminates at a minimum-priority bidder in *candidates*(A, u, M).

Proof. The edges of $M \oplus M'$ form a collection $\mathcal{S}$ of disjoint cycles and paths of positive length.

We begin by arguing that $\mathcal{S}$ does not contain any cycles. Suppose there is a cycle C in $\mathcal{S}$. Let X denote the edges of C that belong to $M \setminus M'$, and let Y denote the edges of C that belong to $M' \setminus M$. Let M'' denote $(M \cup Y) \setminus X$, which is a matching in A since u is unmatched in M and hence does not belong to C. Since M is an MWM of A and $w(M'') = w(M) + w(Y) - w(X)$, we conclude that $w(X) \geq w(Y)$. Let M''' denote $(M' \cup X) \setminus Y$, which is a matching in A'. Since M' is an MWM of A' and $w(M''') = w(M') + w(Y) - w(X)$, we conclude that $w(X) \leq w(Y)$. Thus $w(X) = w(Y)$ and hence $w(M''') =$

$w(M')$, implying that M''' is an MWM of A'. Moreover, since M''' matches the same set of bidders as M', we find that M''' is a greedy MWM of A'. This contradicts the definition of M' since $|M \oplus M'''| < |M \oplus M'|$.

Next we argue that if Q is a path in $\mathcal{S}$, then u is an endpoint of Q. Suppose there is a path Q in $\mathcal{S}$ such that u is not an endpoint of Q. Thus u does not appear on Q since u is unmatched in M. Let X denote the edges of Q that belong to $M \setminus M'$, and let Y denote the edges of Q that belong to $M' \setminus M$. Let M'' denote $(M \cup Y) \setminus X$, which is a matching in A since u does not belong to Q. Since M is an MWM of A and $w(M'') = w(M) + w(Y) - w(X)$, we conclude that $w(X) \geq w(Y)$. Let M''' denote $(M' \cup X) \setminus Y$, which is a matching in A'. Since M' is an MWM of A' and $w(M''') = w(M') + w(Y) - w(X)$, we conclude that $w(X) \leq w(Y)$. Thus $w(X) = w(Y)$ and hence $w(M'') = w(M)$ and $w(M''') = w(M')$, implying that M'' is an MWM of A and M''' is an MWM of A'. Since M is a greedy MWM and hence an MCMWM of A, the set of bidders matched by M is not properly contained in the set of bidders matched by M''; we conclude that $|X| \geq |Y|$. Since M' is a greedy MWM and hence an MCMWM of A', the set of bidders matched by M' is not properly contained in the set of bidders matched by M'''; we conclude that $|X| \leq |Y|$. Thus $|X| = |Y|$, so the length of path Q is even. We consider two cases.

Case 1: The endpoints of Q are items. In this case, M' and M''' match the same set of bidders, and hence M''' is a greedy MWM of A'. This contradicts the definition of M', since Q has positive length and hence $|M \oplus M'''| < |M \oplus M'|$.

Case 2: The endpoints of Q are bidders. Since Q has positive length, one endpoint, call it u_0, is matched in M but not in M', and the other endpoint, call it u_1, is matched in M' but not in M. Since M is a greedy MWM of A and M'' is an MWM of A, we deduce that $\mathit{priority}(u_0) \geq \mathit{priority}(u_1)$. Since M' is a greedy MWM of A' and M''' is an MWM of A', we deduce that $\mathit{priority}(u_0) \leq \mathit{priority}(u_1)$. Thus $\mathit{priority}(u_0) = \mathit{priority}(u_1)$. It follows that $\mathit{priority}(M''') = \mathit{priority}(M')$. Hence M''' is a greedy MWM of A'. This contradicts the definition of M' since $|M \oplus M'''| < |M \oplus M'|$.

From the preceding arguments, we deduce that either $M = M'$ or $M \oplus M'$ corresponds to a positive-length path with u as an endpoint. Equivalently, $M \oplus M'$ is the edge set of a path that has u as an endpoint and may have length zero (i.e., the path may begin and end at u). We claim if the edges of this path are directed away from endpoint u, we obtain a directed path P satisfying the six conditions stated in the lemma. It is easy to see that P satisfies the first four of these conditions. It remains to establish that P satisfies the fifth and sixth conditions.

For the fifth condition, assume that $\mathit{holes}(A, u, M)$ is nonempty. We need to prove that P terminates at an item in $\mathit{holes}(A, u, M)$. Since $\mathit{holes}(A, u, M)$ is nonempty, we deduce that $|M'| = |M| + 1$, and hence that P terminates at some item v. Since P has weight $w(A) - w(A')$, we deduce that v belongs to $\mathit{holes}(A, u, M)$, as required.

For the sixth condition, assume that $\mathit{holes}(A, u, M)$ is empty. We need to prove that P terminates at a minimum-priority bidder in $\mathit{candidates}(A, u, M)$.

Suppose P terminates at some item v. Since P has weight $w(A) - w(A')$, we deduce that v belongs to $\mathit{holes}(A, u, M)$, a contradiction. Thus P terminates at some bidder u'. Since P has weight $w(A) - w(A')$, we deduce that u' belongs to $\mathit{candidates}(A, u, M)$. If u' is not a minimum-priority bidder in $\mathit{candidates}(A, u, M)$, it is easy to argue that M' is not a greedy MWM of A', a contradiction. Thus P terminates at a minimum-priority bidder in $\mathit{candidates}(A, u, M)$. □

Lemma 5.2.7. Let $A = (U, V)$ be a UAP, let M be a greedy MWM of A, let u be a bidder that does not belong to U, let A' denote the UAP $(U + u, V)$, let P be a directed path in $\mathit{digraph}(A, u, M)$ of weight $w(A) - w(A')$ from u to an item in $\mathit{holes}(A, u, M)$, and let M^* denote $\mathit{augment}(A, u, M, P)$. Then M^* is a greedy MWM of A'.

Proof. The definition of $\mathit{augment}(A, u, M, P)$ implies that M^* is an MWM of A'. Let M' denote a greedy MWM of A' minimizing $|M \oplus M'|$. Let U' denote the set of bidders in A matched by M. Since $\mathit{holes}(A, u, M)$ is nonempty, Lemma 5.2.6 implies that the set of bidders in A' matched by M' is $U' + u$. Since M^* is an MWM of A' that also matches the set of bidders $U' + u$, we deduce that M^* is a greedy MWM of A'. □

Lemma 5.2.8. Let $A = (U, V)$ be a UAP, let M be a greedy MWM of A, let u be a bidder that does not belong to U, and let A' denote the UAP $(U + u, V)$. Assume that $\mathit{holes}(A, u, M)$ is empty. Let u' denote a minimum-priority bidder in $\mathit{candidates}(A, u, M)$ (which is nonempty by Lemma 5.2.6), let P be a directed path in $\mathit{digraph}(A, u, M)$ of weight $w(A) - w(A')$ from u

to u', and let M^* denote *augment*(A, u, M, P). Then M^* is a greedy MWM of A'.

Proof. The definition of *augment*(A, u, M, P) implies that M^* is an MWM of A'. Let M' denote a greedy MWM of A' minimizing $|M \oplus M'|$. Let U' denote the set of bidders in A matched by M. Since *holes*(A, u, M) is empty, Lemma 5.2.6 implies that the set of bidders in A' matched by M' is $U' + u - u''$, where u'' is some minimum-priority bidder in *candidates*(A, u, M). It is straightforward to check that M^* has the same weight, cardinality, and priority as M'. Thus M^* is a greedy MWM of A', as required. $\square$

Lemma 5.2.9. Let A and A' be two UAPs such that A' extends A, let M be a greedy MWM of A, and let M' be a greedy MWM of A'. Then $|M'| \geq |M|$.

Proof. Immediate from Lemmas 5.2.7 and 5.2.8. $\square$

5.2.4 Threshold of an Item

In this section, we define the notion of a "threshold" of an item in a UAP. This lays the groundwork for a corresponding IUAP definition in Section 5.3.3. Item thresholds play an important role in our strategyproofness results.

Lemma 5.2.10. Let $A = (U, V)$ be a UAP and let v be an item in V. Let U' be the set of bidders u such that $A + u$ is a UAP and *bid*(u) is of the form $\{(v, x)\}$. Then there is a unique pair of reals (x^*, z^*) such that for any bidder u in U', the following conditions hold, where A' denotes $A + u$, x denotes $w(u, v)$, and z denotes *priority*(u): (1) if $(x, z) > (x^*, z^*)$ then u is matched in every greedy MWM of A'; (2) if $(x,z) < (x^*, z^*)$ then u is not matched in any greedy MWM of A'; (3) if $(x,z) = (x^*, z^*)$ then u is matched in some but not all greedy MWMs of A'.

Proof. Let M be a greedy MWM of A, let W denote $w(M)$, and let Z denote $\mathit{priority}(M)$. Let $\mathcal{M}$ denote the set of matchings of A' that do not match v, let $\mathcal{M}'$ denote the maximum-weight elements of $\mathcal{M}$, let $\mathcal{M}''$ denote the maximum-cardinality elements of $\mathcal{M}'$, let $\mathcal{M}'''$ denote the maximum-priority elements of $\mathcal{M}''$, and observe that there is a unique pair of reals (W', Z') such that any matching M' in $\mathcal{M}'''$ has weight W' and priority Z'. It is straightforward to verify that the unique choice of (x^*, z^*) satisfying the conditions stated in the lemma is $(W - W', Z - Z')$. □

For any UAP $A = (U, V)$ and any item v in V, we define the unique pair (x^*, z^*) of Lemma 5.2.10 as *threshold*(A, v).

5.3 Iterated Unit-Demand Auctions with Priorities

In this section, we formally define the notion of an iterated unit-demand auction with priorities (IUAP). An IUAP allows the bidders, called "multibidders" in this context, to have a sequence of unit-demand bids instead of a single unit-demand bid. In Section 5.3.1, we define a mapping from an IUAP to a UAP by describing an algorithm that generalizes the DA algorithm, and we establish Lemma 5.3.5 that is useful for analyzing the matching produced by Algorithm 5.2 of Section 5.4. Lemma 5.3.5 is used to establish weak stability (Lemmas 5.4.1, 5.4.2, and 5.4.3) and Pareto-optimality (Lemma 5.4.4). In Section 5.3.3, we define the threshold of an item in an IUAP and we establish Lemma 5.3.8, which plays a key role in establishing our strategyproofness results. We start with some useful definitions.

A *multibidder t for a set of items V* is a pair (σ, z) where z is a real priority and σ is a sequence of bidders for V such that all the bidders in σ have distinct IDs and a common priority z. We define $\mathit{priority}(t)$ as z. For any integer i such that $1 \leq i \leq |\sigma|$, we define $\mathit{bidder}(t, i)$ as the bidder $\sigma(i)$. For any integer i such that $0 \leq i \leq |\sigma|$, we define $\mathit{bidders}(t, i)$ as $\{\mathit{bidder}(t, j) \mid 1 \leq j \leq i\}$. We define $\mathit{bidders}(t)$ as $\mathit{bidders}(t, |\sigma|)$.

An *iterated UAP (IUAP)* is a pair $B = (T, V)$ where V is a set of items and T is a set of multibidders for V. In addition, for any distinct multibidders t and t' in T, the following conditions hold: $\mathit{priority}(t) \neq \mathit{priority}(t')$; if u belongs to $\mathit{bidders}(t)$ and u' belongs to $\mathit{bidders}(t')$, then $\mathit{id}(u) \neq \mathit{id}(u')$. For any IUAP $B = (T, V)$, we define $\mathit{bidders}(B)$ as the union, over all t in T, of $\mathit{bidders}(t)$.

5.3.1 Mapping an IUAP to a UAP

Having defined the notion of an IUAP, we now describe an algorithm that maps a given IUAP to a UAP. Our algorithm generalizes the DA algorithm. In each iteration of the DA algorithm, an arbitrary single man is chosen, and this man reveals his next choice. In each iteration of our algorithm, an arbitrary single multibidder is chosen, and this multibidder reveals its next bid. We prove in Lemma 5.3.4 that, like the DA algorithm, our algorithm is confluent: the output does not depend on the nondeterministic choices made during an execution. We conclude this section by establishing Lemma 5.3.5, which is useful for analyzing the matching produced by Algorithm 5.2 in Section 5.4.1. Lemma 5.3.5 is used to establish weak stability (Lemmas 5.4.1, 5.4.2, and 5.4.3) and Pareto-optimality (Lemma 5.4.4). We start with some useful definitions.

Let A be a UAP (U, V) and let B be an IUAP (T, V). The predicate $\mathit{prefix}(A, B)$ is said to hold if $U \subseteq \mathit{bidders}(B)$ and for any multibidder t in T, $U \cap \mathit{bidders}(t) = \mathit{bidders}(t, i)$ for some i.

A *configuration* C is a pair (A, B) where A is a UAP, B is an IUAP, and $\mathit{prefix}(A, B)$ holds.

Let $C = (A, B)$ be a configuration, where $A = (U, V)$ and $B = (T, V)$, and let u be a bidder in U. Then we define $\mathit{multibidder}(C, u)$ as the unique multibidder t in T such that u belongs to $\mathit{bidders}(t)$.

Let $C = (A, B)$ be a configuration where $A = (U, V)$ and $B = (T, V)$. For any t in T, we define $\mathit{bidders}(C, t)$ as $\{u \in U \mid \mathit{multibidder}(C, u) = t\}$.

Let $C = (A, B)$ be a configuration where $B = (T, V)$. We define $\mathit{ready}(C)$ as the set of all bidders u in $\mathit{bidders}(B)$ such that $\mathit{greedy}(A, \mathit{priority}(u))$ $= 0$ and $u = \mathit{bidder}(t, |\mathit{bidders}(C, t)| + 1)$ where $t = \mathit{multibidder}(C, u)$.

Our algorithm for mapping an IUAP to a UAP is Algorithm 5.1. The input is an IUAP B and the output is a UAP A such that $\mathit{prefix}(A, B)$ holds. The algorithm starts with the UAP consisting of all the items in V but no

Algorithm 5.1 An algorithm that maps a given IUAP to a UAP.

```
Input: An IUAP $B = (T, V)$.
Output: A UAP $A$ that $\textit{prefix}(A, B)$ holds.
1: $A \leftarrow (\emptyset, V)$
2: $C \leftarrow (A, B)$
3: while $\textit{ready}(C)$ is nonempty do
4:     $A \leftarrow A +$ an arbitrary bidder in $\textit{ready}(C)$
5:     $C \leftarrow (A, B)$
6: end while
7: return $A$
```

bidders. At this point, no bidder of any multibidder is "revealed". Then, the algorithm iteratively and chooses an arbitrary "ready" bidder and "reveals" it by adding it to the UAP that is maintained in the program variable A. A bidder u associated with some multibidder $t = (\sigma, z)$ is ready if u is not revealed and for each bidder u' that precedes u in σ, u' is revealed and is not matched in any greedy MWM of A. It is easy to verify that the predicate $\textit{prefix}(A, B)$ is an invariant of the algorithm loop: if a bidder u belonging to a multibidder t is to be revealed at an iteration, and $U \cap \textit{bidders}(t) = \textit{bidders}(t, i)$ for some integer i at the beginning of this iteration, then $U \cap \textit{bidders}(t) = \textit{bidders}(t, i + 1)$ after revealing u, where (U, V) is the UAP that is maintained by the program variable A at the beginning of the iteration. No bidder can be revealed more than once since a bidder cannot be ready after it has been revealed; it follows that the algorithm terminates. We now argue that the output of the algorithm is uniquely determined (Lemma 5.3.4), even though the bidder that is revealed in each iteration is chosen nondeterministically.

For any configuration $C = (A, B)$, we define the predicate $\textit{tail}(C)$ to hold if for any bidder u that is matched in some greedy MWM of A, we have $u = \mathit{bidder}(t, |\mathit{bidders}(C,t)|)$ where t denotes $\mathit{multibidder}(C,u)$.

Lemma 5.3.1. Let $C = (A, B)$ be a configuration where $B = (T, V)$ and assume that $\mathit{tail}(C)$ holds. Then $\mathit{greedy}(A, \mathit{priority}(t)) \leq 1$ for each t in T.

Proof. The claim of the lemma easily follows from the definition of $\mathit{tail}(C)$. □

Lemma 5.3.2. The predicate $\mathit{tail}(C)$ is an invariant of the Algorithm 5.1 loop.

Proof. It is easy to see that $\mathit{tail}(C)$ holds when the loop is first encountered. Now consider an iteration of the loop that takes us from configuration $C = (A, B)$ where $A = (U, V)$ to configuration $C' = (A', B)$ where $A' = (U', V)$. We need to show that $\mathit{tail}(C')$ holds. Let u be a bidder that is matched in some greedy MWM M' of A'. Let u^* denote the bidder that is added to A in line 4, and consider the following three cases.

Case 1: $u = u^*$. Let t denote $\mathit{multibidder}(C', u^*)$. In this case, $|\mathit{bidders}(C,t)| + 1 = |\mathit{bidders}(C',t)|$, so $u^* = \mathit{bidder}(t, |\mathit{bidders}(C',t)|)$, as required.

Case 2: $u \neq u^*$ and $\mathit{priority}(u) \neq \mathit{priority}(u^*)$. Since U' contains U, Lemma 5.2.4 implies that u is matched in some greedy MWM of A. Since C is a configuration and $\mathit{tail}(C)$ holds, we deduce that $u = \mathit{bidder}(t, |\mathit{bidders}(C,t)|)$ where t denotes $\mathit{multibidder}(C,u)$. Since $\mathit{multibidder}(C',u) = \mathit{multibidder}(C,u)$ and $\mathit{bidders}(C',t) = \mathit{bidders}(C,t)$, we conclude that $u = \mathit{bidder}(t, |\mathit{bidders}(C',t)|)$, where t denotes $\mathit{multibidder}(C',u)$, as required.

Case 3: $u \neq u^*$ and $\mathit{priority}(u) = \mathit{priority}(u^*)$. Since u^* belongs to $\mathit{ready}(C)$, we know that $\mathit{greedy}(A, \mathit{priority}(u)) = 0$. Also, since u is not u^*, u belongs to U and we conclude that u is not matched in any greedy MWM of A. Since U' contains U, Lemma 5.2.4 implies that u is not matched in any greedy MWM of A', a contradiction. □

Lemma 5.3.3. Let $C = (A, B)$ be a configuration such that $\mathit{tail}(C)$ holds. Then $\mathit{unique}(A)$ holds.

Proof. Let M and M' be greedy MWMs of A, and let u be a bidder in $\mathit{matched}(M)$. To establish the lemma, it is sufficient to prove that u belongs to $\mathit{matched}(M')$. Let t denote $\mathit{multibidder}(C, u)$ and let z denote $\mathit{priority}(t)$. Since $\mathit{tail}(C)$ holds, we know that $u = \mathit{bidder}(t, |\mathit{bidders}(C, t)|)$. Since u is matched by M and since $\mathit{tail}(C)$ holds, Lemma 5.3.1 implies that $\mathit{greedy}(A, z) = 1$. Thus Lemma 5.2.2 implies that M' matches one priority-z bidder. Since $\mathit{tail}(C)$ holds, this bidder is u. □

Lemma 5.3.4. Let $B = (T, V)$ be an IUAP. Then all executions of Algorithm 5.1 on input B produce the same output.

Proof. Suppose not, and let X_1 and X_2 denote two executions of Algorithm 5.1 on input B that produce distinct output UAPs $A_1 = (U_1, V)$ and $A_2 = (U_2, V)$. Without loss of generality, assume that $|U_1| \geq |U_2|$. Then there is a first iteration of execution X_1 in which the bidder added to A in line 4 belongs to $U_1 \setminus U_2$; let u' denote this bidder. Let $C' = (A', B)$ where $A' = (U', V)$ denote the configuration in program variable C at the start of this iteration, and let

t' denote *multibidder*(C', u'). Let i be the integer such that $u' = \mathit{bidder}(t', i)$. We know that $i > 1$ because it is easy to see that U_2 contains $\mathit{bidder}(t', 1)$. Let u'' denote $\mathit{bidder}(t', i-1)$. Since u' belongs to $\mathit{ready}(C')$, Lemmas 5.3.2 and 5.3.3 imply that u'' is not matched in any greedy MWM of A'. Since U' is contained in U_2, Lemma 5.2.4 implies that u'' is not matched in any greedy MWM of A_2. Let $C_2 = (A_2, B)$ denote the final configuration of execution X_2; thus $\mathit{ready}(C_2)$ is empty and $|\mathit{bidders}(C_2, t')| = i - 1$. By Lemma 5.3.2, we conclude that $\mathit{greedy}(A_2, \mathit{priority}(t')) = 0$, and hence that u'' is contained in $\mathit{ready}(C_2)$, a contradiction. □

For any IUAP B, we define $\mathit{uap}(B)$ as the unique (by Lemma 5.3.4) UAP returned by any execution of Algorithm 5.1 on input B.

We now present a lemma that is used in Section 5.4 to establish weak stability (Lemmas 5.4.1, 5.4.2, and 5.4.3) and Pareto-optimality (Lemma 5.4.4).

Lemma 5.3.5. Let $B = (T, V)$ be an IUAP, let (σ, z) be a multibidder that belongs to T, let $\mathit{uap}(B)$ be (U, V), and let M be a greedy MWM of the UAP (U, V). Then the following claims hold: (1) if $\sigma(k)$ is matched in M for some k, then $\sigma(k') \in U$ if and only if $1 \leq k' \leq k$; (2) if $\sigma(k)$ is not matched in M for any k, then $\sigma(k) \in U$ for $1 \leq k \leq |\sigma|$.

Proof. Since $\mathit{prefix}(A, B)$ and $\mathit{tail}(C)$ hold at the end of Algorithm 5.1 by Lemma 5.3.2, the first claim follows. Since $\mathit{ready}(C)$ is empty at the end of Algorithm 5.1, the second claim follows. □

5.3.2 Hungarian-Based Implementation of Algorithm 5.1

In this section, we briefly discuss how to implement Algorithm 5.1 efficiently and how to compute a greedy MWM of $uap(B)$ by maintaining a greedy MWM of the UAP A. We use the modified incremental Hungarian step of Section 5.2.3 in each iteration of the loop of Algorithm 5.1 to maintain A and a greedy MWM of A, as follows: we maintain dual variables (a price for each item) and a residual graph; the initial greedy MWM is the empty matching; when a bidder u is added to A at line 4, we perform an incremental Hungarian step to process u to update the greedy MWM, the prices, and the residual graph. Since we maintain a greedy MWM of A at each iteration of the loop, it is easy to see that identifying a bidder in $ready(C)$ (or determining that this set is empty) takes $O(|V|)$ time. Thus the whole algorithm can be implemented in $O(|bidders(B)| \cdot |V|^2)$ time.

5.3.3 Threshold of an Item

In this section, we define the threshold of an item in an IUAP and we establish Lemma 5.3.8, which plays a key role in establishing our strategyproofness results. We start with some useful definitions.

For any IUAP B, Lemmas 5.3.2 and 5.3.3 imply that $unique(uap(B))$ holds, and thus that every greedy MWM of $uap(B)$ matches the same set of bidders. We define this set of matched bidders as $winners(B)$. For any IUAP B, we define $losers(B)$ as $U \setminus winners(B)$ where (U, V) is $uap(B)$.

Let $B = (T, V)$ be an IUAP and let $u = (\alpha, \beta, z)$ be a bidder for V.

Then we define the IUAP $B + u$ as follows: if T contains a multibidder t of the form (σ, z) for some sequence of bidders σ, then we define $B + u$ as $(T - t + t', V)$ where $t' = (\sigma', z)$ and σ' is the sequence of bidders obtained by appending u to σ; otherwise, we define $B + u$ as $(T + t, V)$ where $t = (\langle u \rangle, z)$.

Lemma 5.3.6. Let $B = (T, V)$ and $B' = B + u$ be IUAPs. Then $losers(B) \subseteq losers(B')$.

Proof. Let u' be a bidder in $losers(B)$. Thus u' is not matched in any greedy MWM of $uap(B)$. Using Lemma 5.3.4, it is easy to see that $uap(B')$ extends $uap(B)$. Thus Lemma 5.2.4 implies that u' is not matched in any greedy MWM of $uap(B')$, and hence that u' belongs to $losers(B')$. $\square$

Lemma 5.3.7. Let $B = (T, V)$ be an IUAP and let v be an item in V. For $i \in \{1, 2\}$, let $B_i = B + u_i$ be an IUAP where $u_i = (\alpha_i, \{(v, x_i)\}, z_i)$. Let $A_1 = (U_1, V)$ denote $uap(B_1)$ and let $A_2 = (U_2, V)$ denote $uap(B_2)$. Assume that $\alpha_1 \neq \alpha_2$, $z_1 \neq z_2$, and u_1 belongs to $winners(B_1)$. Then the following claims hold: if u_2 belongs to $winners(B_2)$ then $U_1 - u_1 = U_2 - u_2$; if u_2 belongs to $losers(B_2)$ then $U_1 - u_1$ contains $U_2 - u_2$.

Proof. Let B_3 denote the IUAP $B_1 + u_2$, which is equal to $B_2 + u_1$. For the first claim, assume that u_2 belongs to $winners(B_2)$. Using Lemma 5.3.4, it is straightforward to argue that $uap(B_3)$ is equal to $A_1 + u_2 = (U_1 + u_2, V)$ and is also equal to $A_2 + u_1 = (U_2 + u_1, V)$. Since u_1 belongs to U_1 and u_2 belongs to U_2, we conclude that $U_1 - u_1 = U_2 - u_2$, as required.

For the second claim, assume that u_2 belongs to $losers(B_2)$. Suppose $(x_1, z_1) < (x_2, z_2)$. Then Lemmas 5.2.3 and 5.3.4 imply that u_2 belongs to *winners*(B_3). Since u_2 belongs to *losers*(B_2), Lemma 5.3.6 implies that u_2 belongs to *losers*$(B_2 + u_1)$ = *losers*(B_3), a contradiction. Since $z_1 \neq z_2$, we conclude that $(x_1, z_1) > (x_2, z_2)$. Then, Lemma 5.3.4 implies that *uap*$(B_3) =$ *uap*$(B_1) + u_2 = (U_1 + u_2, V)$. Since Lemma 5.3.4 also implies that *uap*(B_3) extends *uap*(B_2), it follows that $U_1 + u_2$ contains U_2, and hence that U_1 contains $U_2 - u_2$. Since u_1 does not belong to U_2, we conclude that $U_1 - u_1$ contains $U_2 - u_2$, as required. □

We are now ready to define the threshold of an item in an IUAP, and to state Lemma 5.3.8. In Section 5.4, Lemma 5.3.8 plays an important role in establishing that our SMIW mechanism is strategyproof (Lemma 5.4.6). The proof of Lemma 5.3.8 is provided in Section 5.3.3.

Let $B = (T, V)$ be an IUAP and let v be an item in V. By Lemma 5.3.7, there is a unique subset U of *bidders*(B) such that the following condition holds: for any IUAP $B' = B + u$ where u is of the form $(\alpha, \{(v, x)\}, z)$ and u belongs to *winners*(B'), *uap*(B') is equal to $(U + u, V)$. We define *uap*(B, v) as the UAP (U, V), and we define *threshold*(B, v) as *threshold*$($*uap*$(B, v), v)$.

Lemma 5.3.8. Let $B = (T, V)$ be an IUAP, let $t = (\sigma, z)$ be a multibidder that belongs to T, and let B' denote the IUAP $(T - t, V)$. Suppose that $(\sigma(k), v)$ is matched in some greedy MWM of *uap*(B) for some k. Then

$$(w(\sigma(k), v), z) \geq \textit{threshold}(B', v). \tag{5.1}$$

Furthermore, for each k' and v' such that $1 \leq k' < k$ and v' belongs to *items*$(\sigma(k'))$, we have

$$(w(\sigma(k'), v'), z) < \textit{threshold}(B', v'). \tag{5.2}$$

Proof of Lemma 5.3.8

The purpose of this section is to prove Lemma 5.3.8. We do so by establishing a stronger result, namely Lemma 5.3.16 below. We start with a useful definition.

For any IUAP B, we define $\textit{priorities}(B)$ as $\{z \mid u \in \textit{winners}(B)$ and $\textit{priority}(u) = z\}$.

Lemma 5.3.9. Let $B = (T, V)$ and $B' = B + u = (T', V)$ be IUAPs, let Z denote $\textit{priorities}(B)$, let Z' denote $\textit{priorities}(B')$, and let z denote $\textit{priority}(u)$. Then $|Z'| \geq |Z|$ and $Z' \subseteq Z + z$.

Proof. Consider running Algorithm 5.1 on input B', where we avoid selecting bidder u from $\textit{ready}(C)$ unless it is the only bidder in $\textit{ready}(C)$. (By Lemma 5.3.4, the final output is the same regardless of which bidder is selected from $\textit{ready}(C)$ at each iteration.) If u never enters $\textit{ready}(C)$, then $\textit{uap}(B') = \textit{uap}(B)$, and so $Z' = Z$, and the claim of the lemma holds.

Now suppose that u enters $\textit{ready}(C)$ at some point. Let $A = (U, V)$ denote the UAP at the start of the iteration in which u is selected from $\textit{ready}(C)$. Then A is equal to $\textit{uap}(B)$, and we deduce that $\textit{uap}(B')$ extends $\textit{uap}(B)$. Lemma 5.3.1 implies that every greedy MWM of $A = \textit{uap}(B)$ (resp., $\textit{uap}(B')$) matches exactly one bidder of each priority in Z (resp., Z'). Then, since $\textit{uap}(B')$ extends $\textit{uap}(B)$, Lemma 5.2.9 implies that $|Z'| \geq |Z|$. Furthermore, letting U' denote the set of all bidders u' in $\mathit{bidders}(B)$ such that $\mathit{priority}(u')$ does not belong to $Z + z$, we deduce that U' is contained in $\mathit{losers}(B) = U \setminus \mathit{winners}(B)$. Then Lemma 5.3.6 implies that no bidder in U' is matched in any greedy MWM of $\mathit{uap}(B')$, and thus $Z' \subseteq Z + z$. $\square$

Lemma 5.3.10. Let $A = (U, V)$ and $A' = A + u$ be UAPs, and let v be an item in V. Then $\mathit{threshold}(A, v) \leq \mathit{threshold}(A', v)$.

Proof. Assume for the sake of contradiction that $\mathit{threshold}(A, v) > \mathit{threshold}(A', v)$. Then there exists a bidder u' such that u' does not belong to $U + u$, $\mathit{bid}(u') = \{(v, x)\}, \mathit{priority}(u') = z$, and

$$\mathit{threshold}(A', v) < (x, z) < \mathit{threshold}(A, v).$$

Since $(x, z) < \mathit{threshold}(A, v)$, Lemma 5.2.10 implies that u' is not matched in any greedy MWM of $A + u'$. Thus Lemma 5.2.4 implies that u' is not matched in any greedy MWM of $A' + u'$. On the other hand, since $\mathit{threshold}(A', v) < (x, z)$, Lemma 5.2.10 implies that u' is matched in every greedy MWM of $A' + u'$, a contradiction. $\square$

Lemma 5.3.11. Let $B = (T, V)$ and $B' = B + u$ be IUAPs where $u = (\alpha, \{(v, x)\}, z)$, v is an item in V, and z does not belong to $\mathit{priorities}(B)$. If u belongs to $\mathit{winners}(B')$, then $(x, z) > \mathit{threshold}(B, v)$. If u belongs to $\mathit{losers}(B')$, then $(x, z) < \mathit{threshold}(B, v)$.

Proof. First, assume that u belongs to $\mathit{winners}(B')$. Thus u is matched in every greedy MWM of $\mathit{uap}(B')$, which is equal to $\mathit{uap}(B, v) + u$ by definition.

Lemma 5.2.10 implies that $(x, z) > \mathit{threshold}(\mathit{uap}(B, v), v) = \mathit{threshold}(B, v)$, as required.

Now assume that u belongs to $\mathit{losers}(B')$. Thus u is not matched in any greedy MWM of $\mathit{uap}(B')$. Define U so that $\mathit{uap}(B') = (U + u, V)$, and let A denote the UAP (U, V). Lemma 5.2.10 implies that $(x, z) < \mathit{threshold}(A, v)$. Lemma 5.3.7 implies that $\mathit{uap}(B, v) + u$ extends $\mathit{uap}(B')$, and hence that $\mathit{uap}(B, v)$ extends A. Lemma 5.3.10 therefore implies that

$$\mathit{threshold}(A, v) \leq \mathit{threshold}(\mathit{uap}(B, v), v) = \mathit{threshold}(B, v).$$

Thus $(x, z) < \mathit{threshold}(B, v)$, as required. □

Lemma 5.3.12. Let $B = (T, V)$ and $B' = B + u$ be IUAPs, and let v be an item in V. Then $\mathit{threshold}(B, v) \leq \mathit{threshold}(B', v)$.

Proof. Let (x, z) denote $\mathit{threshold}(B, v)$, let (x', z') denote $\mathit{threshold}(B', v)$, and assume for the sake of contradiction that $(x, z) > (x', z')$.

Let u' be a bidder $(\alpha, \{(v, x)\}, z'')$ such that z'' does not belong to $\mathit{priorities}(B) + \mathit{priority}(u)$, $z > z''$, and $(x, z'') > (x', z')$. Let B'' denote $B + u'$ and let B''' denote $B' + u'$. Since z'' does not belong to $\mathit{priorities}(B)$, we deduce that u' belongs to either $\mathit{winners}(B'')$ or $\mathit{losers}(B'')$. Then, by Lemma 5.3.11, u' belongs to $\mathit{losers}(B'')$, and hence by Lemma 5.3.6, u' belongs to $\mathit{losers}(B''')$. On the other hand, since z'' does not belong to $\mathit{priorities}(B) + \mathit{priority}(u)$, Lemma 5.3.9 implies that z'' does not belong to $\mathit{priorities}(B')$, and we deduce that u' belongs to either $\mathit{winners}(B''')$ or $\mathit{losers}(B''')$. Then, Lemma 5.3.11

implies that u' belongs to *winners*(B'''), a contradiction. □

Lemma 5.3.13. Let $B = (T, V)$ and $B' = B + u$ be IUAPs where u belongs to *losers*(B'), and let v be an item in V. Then *threshold*$(B', v) =$ *threshold*(B, v).

Proof. Suppose not. Then by Lemma 5.3.12, we have *threshold*$(B, v) <$ *threshold*(B', v). Let z denote *priority*(u). Since $B' = B + u$ and u belongs to *losers*(B'), we deduce that z does not belong to *priorities*(B). Since u belongs to *losers*(B'), we deduce that z does not belong to *priorities*(B'). Hence Lemma 5.3.9 implies that *priorities*$(B') =$ *priorities*(B).

Let B'' denote $B + u'$ where $u' = (\alpha, \{(v, x')\}, z')$ is a bidder such that z' does not belong to *priorities*$(B) + z$ and *threshold*$(B, v) < (x', z') <$ *threshold*(B', v).

Let B''' denote $B' + u'$. Since z' does not belong to *priorities*$(B) + z$, Lemma 5.3.9 implies that z' does not belong to *priorities*(B'), and we deduce that u' belongs to either *winners*(B''') or *losers*(B'''). Since $(x', z') <$ *threshold*(B', v), Lemma 5.3.11 implies that u' belongs to *losers*(B'''). Hence Lemma 5.3.9 implies that *priorities*$(B''') =$ *priorities*(B'). Since we have established above that *priorities*$(B') =$ *priorities*(B), we deduce that *priorities*(B''') $=$ *priorities*(B).

Since z' does not belong to *priorities*(B), we deduce that u' belongs to either *winners*(B'') or *losers*(B''). Since $(x', z') >$ *threshold*(B, v), Lemma 5.3.11 implies that u' belongs to *winners*(B'') and hence z' belongs to *priorities*(B''). We consider two cases.

Case 1: $|$*priorities*$(B'')| \leq |$*priorities*$(B)|$. Lemma 5.3.9 implies that there exists a real z'' in *priorities*(B) that does not belong to *priorities*(B''). Since z does not belong to *priorities*(B), we have $z \neq z''$. Since $B''' = B'' + u$ and $z \neq z''$, Lemma 5.3.9 implies that z'' does not belong to *priorities*(B'''), a contradiction since *priorities*$(B''') =$ *priorities*(B).

Case 2: $|$*priorities*$(B'')| > |$*priorities*$(B)|$. Since *priorities*$(B''') =$ *priorities*(B), we deduce that $|$*priorities*$(B'')| > |$*priorities*$(B''')|$. Since $B''' = B'' + u$, Lemma 5.3.9 implies that $|$*priorities*$(B''')| \geq |$*priorities*$(B'')|$, a contradiction. □

Lemma 5.3.14. Let $B = (T, V)$ and $B' = B + u$ be IUAPs where $u = (\alpha, \beta, z)$ and z does not belong to *priorities*(B), and let v be an item in V. Assume that (v, x) belongs to β, and that *threshold*$(B, v) < (x, z)$. Then u belongs to *winners*(B').

Proof. Suppose not. Let $A' = (U', V)$ denote *uap*(B'). Since z does not belong to *priorities*(B), we deduce that u belongs to U'. Thus u belongs to $U' \setminus$ *winners*$(B') =$ *losers*(B'), and so *threshold*$(B', v) =$ *threshold*(B, v) by Lemma 5.3.13.

Let B'' denote $B' + u'$ where $u' = (\alpha, \{(v, x)\}, z')$ is a bidder such that z' does not belong to *priorities*$(B) + z$, *threshold*$(B, v) < (x, z')$, and $z' < z$. Since z' does not belong to *priorities*$(B) + z$, we deduce that u' belongs to either *winners*(B'') or *losers*(B''). Then, by Lemma 5.3.11, u' belongs to *winners*(B''). Let $A'' = (U'', V)$ denote *uap*(B''), and let M be a greedy MWM of A''. Since u' belongs to *winners*(B''), the edge (u', v) belongs to M. Since u belongs to *losers*(B'), Lemma 5.3.6 implies that u belongs to *losers*(B''), and hence that u is unmatched in M. By Lemma 5.2.3, we find that $(x, z) < (x, z')$ and hence $z < z'$, a contradiction. □

Lemma 5.3.15. Let $B = (T, V)$ and $B_0 = B + u$ be IUAPs where $u = (\alpha, \beta, z)$, z does not belong to *priorities*(B), and $\beta = \{(v_1, x_1), \ldots, (v_k, x_k)\}$. Assume that $(x_i, z) <$ *threshold*(B, v_i) holds for all i such that $1 \leq i \leq k$. Then u belongs to *losers*(B_0).

Proof. Suppose not. Since z does not belong to *priorities*(B), we deduce that u belongs to *winners*(B_0), and hence that z belongs to *priorities*(B_0).

For i ranging from 1 to k, let B_i denote the IUAP $B_{i-1} + u_i$ where $u_i = (\alpha_i, \{(v_i, x_i)\}, z_i)$ and z_i is a real number satisfying the following conditions: z_i does not belong to *priorities*(B_{i-1}); $z < z_i$; $(x_i, z_i) <$ *threshold*(B, v_i). Since z_i does not belong to *priorities*(B_{i-1}), we deduce that u_i belongs to either *winners*(B_i) or *losers*(B_i) for $1 \leq i \leq k$. Then, by Lemmas 5.3.11 and 5.3.12, we deduce that u_i belongs to *losers*(B_i) for $1 \leq i \leq k$. By repeated application of Lemma 5.3.9, we find that *priorities*(B_i) = *priorities*(B_0) for $1 \leq i \leq k$, and hence that z belongs to *priorities*(B_k).

We claim that u belongs to *winners*(B_k). To prove this claim, let t denote the unique multibidder in B_k for which *priority*(t) = *priority*(u). Let ℓ denote $|$*bidders*$(t)|$, and observe that $u =$ *bidder*(t, ℓ). Furthermore, since z does not belong to *priorities*(B), we deduce that *bidder*(t, i) belongs to *losers*(B) for $1 \leq i < \ell$. By repeated application of Lemma 5.3.6, we deduce that *bidder*(t, i) belongs to *losers*(B_k) for $1 \leq i < \ell$. Since z belongs to *priorities*(B_k), the claim follows.

Let M denote a greedy MWM of $\mathit{uap}(B_k)$. Since u belongs to $\mathit{winners}(B_k)$, there is a unique integer i, $1 \leq i \leq k$, such that M contains edge (u, v_i). Let i denote this integer. Since z_i does not belong to $\mathit{priorities}(B_k)$, we know that u_i belongs to $\mathit{losers}(B_k)$ and hence that u_i is not matched in any greedy MWM of $\mathit{uap}(B_k)$. By Lemma 5.2.3, we deduce that $(x_i, z_i) < (x_i, z)$. Hence $z_i < z$, contradicting the definition of z_i. $\square$

Lemma 5.3.16. Let $B_0 = (T, V)$ be an IUAP, let z be a real that is not equal to the priority of any multibidder in T, let k be a nonnegative integer, and for i ranging from 1 to k, let B_i denote the IUAP $B_{i-1} + u_i$, where $\mathit{priority}(u_i) = z$. Let I denote the set of all integers i in $\{1, \ldots, k\}$ such that there exists an item v in V for which $(w(u_i, v), z) > \mathit{threshold}(B_0, v)$. If I is empty, then z does not belong to $\mathit{priorities}(B_k)$. Otherwise, u_j belongs to $\mathit{winners}(B_k)$, where j denotes the minimum integer in I.

Proof. If I is empty, then by repeated application of Lemmas 5.3.13 and 5.3.15, we find that u_i belongs to $\mathit{losers}(B_i)$ for $1 \leq i \leq k$. By repeated application of Lemma 5.3.6, we deduce that u_i belongs to $\mathit{losers}(B_k)$ for $1 \leq i \leq k$. It follows that z does not belong to $\mathit{priorities}(B_k)$, as required.

Now assume that I is nonempty, and let j denote the minimum integer in I. Arguing as in the preceding paragraph, we find that z does not belong to $\mathit{priorities}(B_{j-1})$. By repeated application of Lemma 5.3.13, we deduce that $\mathit{threshold}(B_{j-1}, v) = \mathit{threshold}(B_0, v)$ for all items v in V. Thus Lemma 5.3.14 implies that u_j belongs to $\mathit{winners}(B_j)$. Then, since $u_{j+1}, \ldots, u_k$ all have the same priority as u_j, it is easy to argue by Lemma 5.3.4 that $\mathit{uap}(B_k) =$

$uap(B_j)$, and hence u_j belongs to $winners(B_k)$, as required. □

Proof of Lemma 5.3.8. It is easy to see that the claims of the lemma follow from Lemma 5.3.16. □

5.4 Stable Marriage with Indifferences

The *stable marriage model with incomplete and weak preferences (SMIW)* involves a set P of men and a set Q of women. The preference relation of each man p in P is specified as a binary relation $\succeq_p$ over $Q + \varnothing$ that satisfies transitivity and totality, where $\varnothing$ denotes being unmatched. Similarly, the preference relation of each woman q in Q is specified as a binary relation $\succeq_q$ over $P + \varnothing$ that satisfies transitivity and totality, where $\varnothing$ denotes being unmatched. To allow indifferences, the preference relations are not required to satisfy antisymmetry. We will use $\succ_p$ and $\succ_q$ to denote the asymmetric part of $\succeq_p$ and $\succeq_q$, respectively.

A matching is a function μ from P to $Q + \varnothing$ such that for any woman q in Q, there exists at most one man p in P for which $\mu(p) = q$. Given a matching μ and a woman q in Q, we denote

$$\mu(q) = \begin{cases} p & \text{if } \mu(p) = q \\ \varnothing & \text{if there is no man } p \text{ in } P \text{ such that } \mu(p) = q. \end{cases}$$

A matching μ is *individually rational* if for any man p in P and woman q in Q such that $\mu(p) = q$, we have $q \succeq_p \varnothing$ and $p \succeq_q \varnothing$. A pair (p, q') in

$P \times Q$ is said to form a *strongly blocking pair* for a matching μ if $q' \succ_p \mu(p)$ and $p \succ_{q'} \mu(q')$. A matching is *weakly stable* if it is individually rational and does not admit a strongly blocking pair.

For any matching μ and μ', we say that the binary relation $\mu \succeq \mu'$ holds if for every man p in P and woman q in Q, we have $\mu(p) \succeq_p \mu'(p)$ and $\mu(q) \succeq_q \mu'(q)$. We let $\succ$ denote the asymmetric part of $\succeq$. We say that a matching μ *Pareto-dominates* another matching μ' if $\mu \succ \mu'$. We say that a matching is *Pareto-optimal* if it is not Pareto-dominated by any other matching. A matching is *Pareto-stable* if it is Pareto-optimal and weakly stable.

A *mechanism* is an algorithm that, given $(P, Q, (\succeq_p)_{p \in P}, (\succeq_q)_{q \in Q})$, produces a matching μ. A mechanism is said to be *strategyproof (for the men)* if for any man p in P expressing preference $\succeq'_p$ instead of his true preference $\succeq_p$, we have $\mu(p) \succeq_p \mu'(p)$, where μ and μ' are the matchings produced by the mechanism given $\succeq_p$ and $\succeq'_p$, respectively, when all other inputs are fixed.

By introducing extra men or women who prefer being unmatched to being matched with any potential partner, we may assume without loss of generality that the number of men is equal to the number of women. So, $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_n\}$.

5.4.1 Algorithm 5.2

The computation of a matching for SMIW is shown in Algorithm 5.2. We construct an item for each woman in line 4, and a multibidder for each man in line 13 by examining the tiers of preferences of the men and the utilities of the women. Together with dummy items constructed in line 8, this forms an IUAP, from which we obtain a UAP and a greedy MWM M_0. Using Lemma 5.3.5, we argue that for any man p_i, exactly one of the bidders associated with p_i is matched in M_0; see the proof of Lemma 5.4.1. Finally, in line 18, we use M_0 to determine the match of a man p_i as follows, where u denotes the unique bidder associated with p_i that is matched in M_0: if u is matched in M_0 to the item corresponding to a woman q_j, then we match p_i to q_j; otherwise, u is matched to a dummy item in M_0, and we leave p_i unmatched.

In Lemma 5.4.2, we prove individually rationality by arguing that the dummy items ensure that no man or woman is matched to an unacceptable partner. In Lemma 5.4.3, we prove weak stability using the properties of a greedy MWM. In Lemmas 5.4.4 and 5.4.5, we prove Pareto-optimality by showing that any matching that Pareto-dominates the output matching induces another MWM that contradicts the greediness of the MWM produced by the algorithm. In Lemma 5.4.6, we establish two properties of IUAP thresholds that are used to show strategyproofness in Theorem 5.4.7.

Lemma 5.4.1. Algorithm 5.2 produces a valid matching.

Proof. First, we show that for any man p_i where $1 \leq i \leq n$, there exists at most one j in $\{1, \ldots, 2n\}$ such that bidder $\sigma_i(k)$ is matched to item v_j in M_0 for some k. For the sake of contradiction, suppose bidder $\sigma_i(k)$ is matched to item v_j and bidder $\sigma_i(k')$ is matched to item $v_{j'}$ in M_0 for some k and k' where $j \neq j'$. By Lemma 5.3.5, we have $k \leq k'$ and $k' \leq k$. Therefore, bidder

Algorithm 5.2 A strategyproof Pareto-stable mechanism for SMIW.

```
Input: An SMIW instance $(P, Q, (\succeq_p)_{p \in P}, (\succeq_q)_{q \in Q})$ such that $|P| = |Q|$.
Output: A Pareto-stable matching $\mu$.
 1: Let $p_0$ denote $\varnothing$.
 2: for all $1 \le j \le n$ do
 3:     Convert the preference relation $\succeq_{q_j}$ of woman $q_j$ into utility function
        $\psi_{q_j} : P + \varnothing \to \mathbb{R}$ that satisfies the followings: $\psi_{q_j}(\varnothing) = 0$; for any $i$ and
        $i'$ in $\{0, 1, \ldots, n\}$, we have $p_i \succeq_{q_j} p_{i'}$ if and only if $\psi_{q_j}(p_i) \ge \psi_{q_j}(p_{i'})$. This
        utility assignment should not depend on the preferences of the men.
 4:     Construct an item $v_j$ corresponding to woman $q_j$.
 5: end for
 6: for all $n < j \le 2n$ do
 7:     Let $q_j$ denote $\varnothing$.
 8:     Construct a dummy item $v_j$ corresponding to $q_j$.
 9: end for
10: for all $1 \le i \le n$ do
11:     Partition the set $\{1, \ldots, n\} \cup \{n + i\}$ of woman indices into tiers
        $\tau_i(1), \ldots, \tau_i(K_i)$ according to the preference relation of man $p_i$, such that
        for any $j$ in $\tau_i(k)$ and $j'$ in $\tau_i(k')$, we have $q_j \succeq_{p_i} q_{j'}$ if and only if $k \le k'$.
12:     For $j$ in $\{1, \ldots, n\} \cup \{n + i\}$, denote tier number $\kappa_i(q_j)$ as the unique
        $k$ such that $j$ in $\tau_i(k)$.
13:     Construct a multibidder $t_i = (\sigma_i, z_i)$ with priority $z_i = i$ corresponding
        to man $p_i$. The multibidder $t_i$ has $K_i$ bidders. For each bidder $\sigma_i(k)$ we
        define $items(\sigma_i(k))$ as $\{v_j \mid j \in \tau_i(k)\}$ and $w(\sigma_i(k), q_j)$ as $\psi_{q_j}(p_i)$, where
        $\psi_{q_{n+i}}(p_i)$ is defined to be 0.
14: end for
15: $(T, V) = (\{t_i \mid 1 \le i \le n\}, \{v_j \mid 1 \le j \le 2n\})$.
16: $(U, V) = uap(T, V)$.
17: Compute a greedy MWM $M_0$ of UAP $(U, V)$ as described in Section 5.2.3.
18: return the matching $\mu$ such that for all $1 \le i \le n$ and $1 \le j \le 2n$, we
    have $\mu(p_i) = q_j$ if and only if $\sigma_i(k)$ is matched to item $v_j$ in $M_0$ for some
    $k$.
```

$\sigma_i(k) = \sigma_i(k')$ is matched in M_0 to both v_j and $v_{j'}$, which is a contradiction.

Next, we show that for any man p_i where $1 \le i \le n$, there exists at least one j in $\{1, \ldots, 2n\}$ such that bidder $\sigma_i(k)$ is matched to item v_j in M_0 for some

k. For the sake of contradiction, suppose bidder $\sigma_i(k)$ is unmatched in M_0 for all k. Let j denote $n+i$ and let k denote $\kappa_i(q_j)$. By Lemma 5.3.5, the set U contains bidder $\sigma_i(k)$. Since both bidder $\sigma_i(k)$ and item v_j are unmatched by M_0, adding the pair $(\sigma_i(k), v_j)$ to M_0 gives a matching of (U, V) with the same weight and larger cardinality. This contradicts the fact that M_0 is a greedy MWM of (U, V).

This shows that $\mu(p_i)$ is well-defined for all men p_i where $1 \leq i \leq n$. Furthermore, since each item v_j where $1 \leq j \leq n$ is matched to at most one bidder in M_0, each woman q_j is matched to at most one man p_i in μ where $1 \leq i \leq n$. Hence, μ is a valid matching. $\square$

Lemma 5.4.2. Algorithm 5.2 produces an individually rational matching.

Proof. We have shown in Lemma 5.4.1 that μ is a valid matching. Consider man p_i and woman q_j such that $\mu(p_i) = q_j$, where i and j belong to $\{1, \ldots, n\}$. Let k denote $\kappa_i(q_j)$ and let k' denote $\kappa_i(q_{n+i})$. It suffices to show that $k \leq k'$ and $\psi_{q_j}(p_i) \geq 0$.

Since $\mu(p_i) = q_j$, bidder $\sigma_i(k)$ is matched to item v_j in M_0. Since M_0 is an MWM, we have $\psi_{q_j}(p_i) = w(\sigma_i(k), v_j) \geq 0$.

It remains to show that $k \leq k'$. For the sake of contradiction, suppose $k > k'$. Since bidder $\sigma_i(k)$ is matched to item v_j in M_0, by Lemma 5.3.5 the set U contains bidder $\sigma_i(k')$. Since bidder $\sigma_i(k')$ is not matched in M_0, the dummy item v_{n+i} is also not matched in M_0. Hence, adding the pair $(\sigma_i(k'), v_{n+i})$ to M_0 gives a matching in (U, V) with the same weight and larger cardinality. This contradicts the fact that M_0 is a greedy MWM of (U, V). $\square$

Lemma 5.4.3. Algorithm 5.2 produces a weakly stable matching.

Proof. By Lemma 5.4.2, it remains only to show that μ does not admit a strongly blocking pair. Consider man p_i and woman $q_{j'}$, where i and j' belong to $\{1, \ldots, n\}$. We want to show that $(p_i, q_{j'})$ does not form a strongly blocking pair. Let q_j denote $\mu(p_i)$ and let $p_{i'}$ denote $\mu(q_{j'})$, where j belongs to $\{1, \ldots, n\} \cup \{n+i\}$ and i' belongs to $\{0, 1, \ldots, n\}$. It suffices to show that either $\kappa_i(q_j) \leq \kappa_i(q_{j'})$ or $\psi_{q_{j'}}(p_{i'}) \geq \psi_{q_{j'}}(p_i)$. For the sake of contradiction, suppose $\kappa_i(q_j) > \kappa_i(q_{j'})$ and $\psi_{q_{j'}}(p_{i'}) < \psi_{q_{j'}}(p_i)$. Let k denote $\kappa_i(q_j)$ and let k' denote $\kappa_i(q_{j'})$. Since $\sigma_i(k)$ is matched in M_0 and $k' < k$, Lemma 5.3.5 implies that the set U contains bidder $\sigma_i(k')$ and that $\sigma_i(k')$ is unmatched in M_0. We consider two cases.

Case 1: $i' = 0$. Then $\psi_{q_{j'}}(p_i) > \psi_{q_{j'}}(p_{i'}) = 0$. Since neither bidder $\sigma_i(k')$ nor item $v_{j'}$ is matched in M_0, adding the pair $(\sigma_i(k'), v_{j'})$ to M_0 gives a matching of (U, V) with a larger weight. This contradicts the fact that M_0 is an MWM of (U, V).

Case 2: $i' \neq 0$. Since $p_{i'} = \mu(q_{j'})$, there exists k'' such that bidder $\sigma_{i'}(k'')$ is matched to $v_{j'}$ in M_0. Since $\sigma_i(k')$ is unmatched in M_0, the matching $M_0 - (\sigma_{i'}(k''), v_{j'}) + (\sigma_i(k'), v_{j'})$ is a matching of (U, V) with weight $w(M_0) - \psi_{q_{j'}}(p_{i'}) + \psi_{q_{j'}}(p_i)$, which is greater than $w(M_0)$. This contradicts the fact that M_0 is an MWM of (U, V). $\square$

Lemma 5.4.4. Let μ be the matching produced by Algorithm 5.2 and let μ'

be a matching such that $\mu'(p) \succeq_p \mu(p)$ for every man p in P and

$$\sum_{q \in Q} \psi_q(\mu'(q)) \geq \sum_{q \in Q} \psi_q(\mu(q)).$$

Then $\mu(p) \succeq_p \mu'(p)$ for every man p in P and

$$\sum_{q \in Q} \psi_q(\mu'(q)) = \sum_{q \in Q} \psi_q(\mu(q)).$$

Proof. For any i such that $1 \leq i \leq n$, let k_i denote $\kappa_i(\mu(p_i))$ and let k'_i denote $\kappa_i(\mu'(p_i))$.

Below we use μ' to construct an MWM M'_0 of (U, V). We give the construction of M'_0 first, and then argue that M'_0 is an MWM of (U, V). Let M'_0 denote the set of bidder-item pairs $(\sigma_i(k'_i), v_j)$ such that $\mu'(p_i) = q_j$ where i in $\{1, \ldots, n\}$ and j in $\{1, \ldots, n\} \cup \{n + i\}$. It is easy to see that M'_0 is a valid matching. Notice that for any $1 \leq i \leq n$, since $\mu'(p_i) \succeq_{p_i} \mu(p_i)$, we have $k'_i \leq k_i$. So, by Lemma 5.3.5, the set U contains all bidders $\sigma_i(k'_i)$. Hence, M'_0 is a matching of (U, V). Furthermore, it is easy to see that

$$w(M'_0) = \sum_{1 \leq j \leq n} \psi_{q_j}(\mu'(q_j)) \geq \sum_{1 \leq j \leq n} \psi_{q_j}(\mu(q_j)) = w(M_0).$$

Thus M'_0 is an MWM of (U, V), and we have

$$\sum_{1 \leq j \leq n} \psi_{q_j}(\mu'(q_j)) = \sum_{1 \leq j \leq n} \psi_{q_j}(\mu(q_j)).$$

Furthermore, M_0' is an MCMWM of (U, V) because both M_0' and M_0 have cardinality equal to n. Also, M_0' is a greedy MWM of (U, V), because both M_0' and M_0 have priorities equal to $\sum_{1 \le i \le n} z_i$. Hence, for each $1 \le i \le n$, we have $k_i \le k_i'$ by Lemma 5.3.5. Thus, $\mu(p_i) \succeq_{p_i} \mu'(p_i)$ for all $1 \le i \le n$. $\square$

Lemma 5.4.5. Let μ be the matching produced by Algorithm 5.2 and μ' be a matching such that $\mu' \succeq \mu$. Then, $\mu \succeq \mu'$.

Proof. Since $\mu' \succeq \mu$, we have $\mu'(p_i) \succeq_{p_i} \mu(p_i)$ and $\psi_{q_j}(\mu'(q_j)) \ge \psi_{q_j}(\mu(q_j))$ for every i and j in $\{1, \ldots, n\}$. So, by Lemma 5.4.4, we have $\mu(p_i) \succeq_{p_i} \mu'(p_i)$ for every i in $\{1, \ldots, n\}$ and

$$\sum_{1 \le j \le n} \psi_{q_j}(\mu'(q_j)) = \sum_{1 \le j \le n} \psi_{q_j}(\mu(q_j)).$$

Therefore, $\psi_{q_j}(\mu'(q_j)) = \psi_{q_j}(\mu(q_j))$ for every j in $\{1, \ldots, n\}$. This shows that $\mu \succeq \mu'$. $\square$

Lemma 5.4.6. Consider Algorithm 5.2. Suppose $\mu(p_i) = q_j$, where $1 \le i \le n$ and j belongs to $\{1, \ldots, n\} \cup \{n + i\}$. Then, we have

$$(\psi_{q_j}(p_i), i) \ge \textit{threshold}((T - t_i, V), v_j). \tag{5.3}$$

Furthermore, for all j' in $\{1, \ldots, n\} \cup \{n + i\}$ such that $\kappa_i(q_{j'}) < \kappa_i(q_j)$, we have

$$(\psi_{q_{j'}}(p_i), i) < \textit{threshold}((T - t_i, V), v_{j'}). \tag{5.4}$$

Proof. Let k denote $\kappa_i(q_j)$. Since $\mu(p_i) = q_j$, we know that bidder $\sigma_i(k)$

is matched to item v_j in M_0. So, inequality (5.1) of Lemma 5.3.8 implies inequality (5.3), because $w(\sigma_i(k), v_j) = \psi_{q_j}(p_i)$ and $z_i = i$.

Now, suppose $\kappa_i(q_{j'}) < \kappa_i(q_j)$. Let k' denote $\kappa_i(q_{j'})$. Since $k' < k$, inequality (5.2) of Lemma 5.3.8 implies inequality (5.4), because $w(\sigma_i(k'), v_{j'}) = \psi_{q_{j'}}(p_i)$ and $z_i = i$. $\square$

Theorem 5.4.7. Algorithm 5.2 is a strategyproof Pareto-stable mechanism for the stable marriage problem with incomplete and weak preferences (for any fixed choice of utility assignment).

Proof. We have shown in Lemma 5.4.3 that the algorithm produces a weakly stable matching. Moreover, Lemma 5.4.5 shows that the weakly stable matching produced is not Pareto-dominated by any other matching. Hence, the algorithm produces a Pareto-stable matching. It remains to show that the algorithm is a strategyproof mechanism.

Suppose man p_i expresses $\succeq'_{p_i}$ instead of his true preference relation $\succeq_{p_i}$, where $1 \leq i \leq n$. Let μ and μ' be the resulting matchings given $\succeq_{p_i}$ and $\succeq'_{p_i}$, respectively. Let q_j denote $\mu(p_i)$ and let $q_{j'}$ denote $\mu'(p_i)$, where j and j' belong to $\{1, \ldots, n\} \cup \{n+i\}$. Let k denote $\kappa_i(q_j)$ and let k' denote $\kappa_i(q_{j'})$, where $\kappa_i(\cdot)$ denotes the tier number with respect to $\succeq_{p_i}$. It suffices to show that $k \leq k'$. For the sake of contradiction, suppose $k > k'$.

Let (T, V) be the IUAP, let t_i be the multibidder corresponding to man p_i, and let $v_{j'}$ be the item corresponding to woman $q_{j'}$ constructed in the algorithm given input $\succeq_{p_i}$. Since $\mu(p_i) = q_j$, by inequality (5.4) of Lemma 5.4.6, we have

$$(\psi_{q_{j'}}(p_i), i) < \textit{threshold}((T - t_i, V), v_{j'}).$$

Now, consider the behavior of the algorithm when preference relation $\succeq_{p_i}$ is replaced with $\succeq'_{p_i}$. Let (T', V') be the IUAP, let t'_i be the multibidder corresponding to man p_i, and let $v'_{j'}$ be the item corresponding to woman $q_{j'}$ constructed in the algorithm given input $\succeq'_{p_i}$. Since $\mu'(p_i) = q_{j'}$, by inequality (5.3) of Lemma 5.4.6, we have

$$(\psi_{q_{j'}}(p_i), i) \geq \textit{threshold}((T' - t'_i, V'), v'_{j'}).$$

Notice that in Algorithm 5.2, the only part of the IUAP instance that depends on the preferences of man p_i is the multibidder corresponding to man p_i. In particular, we have $T - t_i = T' - t'_i$, $V = V'$, and $v_{j'} = v'_{j'}$. Hence, we get

$$\begin{aligned}(\psi_{q_{j'}}(p_i), i) &< \textit{threshold}((T - t_i, V), v_{j'})\\ &= \textit{threshold}((T' - t'_i, V'), v'_{j'})\\ &\leq (\psi_{q_{j'}}(p_i), i),\end{aligned}$$

which is a contradiction. □

5.5 College Admissions with Indifferences

Our strategyproof mechanism can be generalized to the college admissions model with weak preferences. In this model, students and colleges play the roles of men and women, respectively, and colleges are allowed to be matched with multiple students up to their capacities. We can apply our mechanism for SMIW by transforming each student to a man and each slot of a college to a woman in a standard fashion.

The *college admissions model with weak preferences (CAW)* involves a set P of students and a set Q of colleges. The preference relation of each student p in P is specified as a binary relation $\succeq_p$ over $Q + \varnothing$ that satisfies transitivity and totality, where $\varnothing$ denotes being unmatched. The preference relation of each college q in Q over individual students is specified as a binary relation $\succeq_q$ over $P + \varnothing$ that satisfies transitivity and totality, where $\varnothing$ denotes being unmatched. Each college q in Q has an associated integer capacity $c_q > 0$. We will use $\succ_p$ and $\succ_q$ to denote the asymmetric parts of $\succeq_p$ and $\succeq_q$, respectively.

The colleges' preference relations over individual students can be extended to group preference relations using responsiveness [49]. We say that a transitive and reflexive relation $\succeq'_q$ over the power set 2^P is *responsive to the preference relation* $\succeq_q$ if the following conditions hold: for any $S \subseteq P$ and p in $P \setminus S$, we have $p \succeq_q \varnothing$ if and only if $S + p \succeq'_q S$; for any $S \subseteq P$ and any p and p' in $P \setminus S$, we have $p \succeq_q p'$ if and only if $S + p \succeq'_q S + p$. Furthermore, we say that a relation $\succeq'_q$ is *minimally responsive to the preference relation* $\succeq_q$

if it is responsive to the preference relation $\succeq_q$ and does not strictly contain another relation that is responsive to the preference relation $\succeq_q$.

A *(capacitated) matching* is a function μ from P to $Q + \varnothing$ such that for any college q in Q, there exists at most c_q students p in P for which $\mu(p) = q$. Given a matching μ and a college q in Q, we let $\mu(q)$ denote $\{p \in P \mid \mu(p) = q\}$.

A matching μ is *individually rational* if for any student p in P and college q in Q such that $\mu(p) = q$, we have $q \succeq_p \varnothing$ and $p \succeq_q \varnothing$. A pair (p', q) in $P \times Q$ is said to form a *strongly blocking pair* for a matching μ if $q \succ_{p'} \mu(p')$ and at least one of the following two conditions holds: (1) there exists a student p in P such that $\mu(p) = q$ and $p' \succ_q p$; (2) $|\mu(q)| < c_q$ and $p' \succ_q \varnothing$. A matching is *weakly stable* if it is individually rational and does not admit a strongly blocking pair.

Let $\succeq'_q$ be the group preference associated with college q in Q. For any matching μ and μ', we say that the binary relation $\mu \succeq \mu'$ holds if for every student p in P and college q in Q, we have $\mu(p) \succeq_p \mu'(p)$ and $\mu(q) \succeq'_q \mu'(q)$. We let $\succ$ denote the asymmetric part of $\succeq$. We say that a matching μ *Pareto-dominates* another matching μ' if $\mu \succ \mu'$. We say that a matching is *Pareto-optimal* if it is not Pareto-dominated by any other matching. A matching is *Pareto-stable* if it is Pareto-optimal and weakly stable.

A *mechanism* is an algorithm that, given $(P, Q, (\succeq_p)_{p\in P}, (\succeq_q)_{q\in Q}, (c_q)_{q\in Q})$, produces a matching μ. A mechanism is said to be *strategyproof (for the students)* if for any student p in P expressing preference $\succeq'_p$ instead of their true preference $\succeq_p$, we have $\mu(p) \succeq_p \mu'(p)$, where μ and μ' are the matchings produced by the mechanism given $\succeq_p$ and $\succeq'_p$, respectively, when all other inputs are fixed.

Without loss of generality, we may assume that the number of students equals the total capacity of the colleges. So, $P = \{p_i\}_{1 \le i \le |P|}$ and $Q = \{q_j\}_{1 \le j \le |Q|}$ such that $|P| = \sum_{1 \le j \le |Q|} c_{q_j}$.

5.5.1 Algorithm 5.3

The computation of a matching for CAW is shown in Algorithm 5.3. We transform each student to a man in line 1, and each slot of a college to a woman in line 2. This forms an SMIW. Using this SMIW, we produce a matching by invoking Algorithm 5.2 in lines 8 and 9.

The following results are analogues of that in Section 5.4.1

Lemma 5.5.1. Algorithm 5.3 produces an individually rational matching.

Proof. It is easy to see that μ satisfies the capacity constraints because each college q_j is associated with c_{q_j} women q'_{jk} and each woman can be matched with at most one man in μ_0 by Lemma 5.4.1.

The individual rationality of μ follows from the individual rationality of μ_0. Let p_i in P and q_j in Q such that $\mu(p_i) = q_j$. Then $\mu_0(p'_i) = q'_{jk}$ for some k. By Lemma 5.4.2, we have $q'_{jk} \succeq_{p'_i} \varnothing$ and $p'_i \succeq_{q'_{jk}} \varnothing$. Hence, $q_j \succeq_{p_i} \varnothing$ and $p_i \succeq_{q_j} \varnothing$. $\square$

Lemma 5.5.2. Algorithm 5.3 produces a weakly stable matching.

Algorithm 5.3 A strategyproof Pareto-stable mechanism for CAW.

```
Input: An CAW instance $(P, Q, (\succeq_p)_{p\in P}, (\succeq_q)_{q\in Q}, (c_q)_{q\in Q})$ such that $|P| = \sum_{q\in Q} c_q$.
Output: A Pareto-stable matching $\mu$.
1: For each $1 \le i \le |P|$, construct man $p'_i$ corresponding to student $p_i$.
2: For each $1 \le j \le |Q|$, construct women $q'_{j1}, \ldots, q'_{jc}$ corresponding to college $q_j$ with capacity $c = c_{q_j}$.
3: $(P', Q') = (\{p'_i \mid 1 \le i \le |P|\}, \{q'_{jk} \mid 1 \le j \le |Q| \text{ and } 1 \le k \le c_{q_j}\})$.
4: Let $p_0$ denote $\varnothing$. Let $p'_0$ denote $\varnothing$.
5: Let $q_0$ denote $\varnothing$. Let $q'_{00}$ denote $\varnothing$.
6: For each $1 \le i \le |P|$, define the preference relation $\succeq_{p'_i}$ over $Q' + q'_{00}$ for man $p'_i$ using the preference relation of his corresponding student, such that $q'_{jk} \succeq_{p'_i} q'_{j'k'}$ if and only if $q_j \succeq_{p_i} q_{j'}$.
7: For each $1 \le j \le |Q|$ and $1 \le k \le c_{q_j}$, define the preference relation $\succeq_{q'_{jk}}$ over $P' + p'_0$ for woman $q'_{jk}$ using the preference relation of her corresponding college, such that $p'_i \succeq_{q'_{jk}} p'_{i'}$ if and only if $p_i \succeq_{q_j} p_{i'}$.
8: Compute matching $\mu_0$ for SMIW $(P', Q', (\succeq_{p'})_{p'\in P'}, (\succeq_{q'})_{q'\in Q'})$ using Algorithm 5.2, where we require the utility functions associated with the same college to be the same.
9: return the matching $\mu$, such that for all $1 \le i \le |P|$ and $0 \le j \le |Q|$, we have $\mu(p_i) = q_j$ if and only if $\mu_0(p'_i) = q'_{jk}$ for some $k$.
```

Proof. By Lemma 5.5.1, it remains only to show that μ does not admit a strongly blocking pair. Consider student $p_{i'}$ in P and college q_j in Q. In what follows, we use the weak stability of μ_0 to show that $(p_{i'}, q_j)$ does not form a strongly blocking pair.

Let $q'_{j'k'}$ denote $\mu_0(p'_{i'})$. It is possible that $q'_{j'k'} = \varnothing$, in which case $j' = k' = 0$. For $1 \le k \le c_{q_j}$, let p'_{i_k} denote $\mu_0(q'_{jk})$, where p'_{i_k} belongs to $P' + p'_0$. By Lemma 5.4.3, for any $1 \le k \le c_{q_j}$, either $q'_{j'k'} \succeq_{p'_{i'}} q'_{jk}$ or $p'_{i_k} \succeq_{q'_{jk}} p'_{i'}$, for otherwise $(p'_{i'}, q'_{jk})$ forms a strongly blocking pair.

Suppose $q'_{j'k'} \succeq_{p'_{i'}} q'_{jk}$ for some $1 \le k \le c_{q_j}$. Then $q_{j'} \succeq_{p_{i'}} q_j$, and hence

$(p_{i'}, q_j)$ does not form a strongly blocking pair.

Otherwise, $p'_{i_k} \succeq_{q'_{jk}} p'_{i'}$ for all $1 \leq k \leq c_{q_j}$. Then $p_{i_k} \succeq_{q_j} p_{i'}$ for all $1 \leq k \leq c_{q_j}$. In particular, we have $p_{i_k} \succeq_{q_j} p_{i'}$ for all students p_{i_k} in P such that $\mu(p_{i_k}) = q_j$. Furthermore, if $|\mu(q_j)| < c_{q_j}$, then $p_{i_k} = \varnothing$ for some $1 \leq k \leq c_{q_j}$. Hence $\varnothing \succeq_{q_j} p_{i'}$. It follows that $(p_{i'}, q_j)$ does not form a strongly blocking pair. $\square$

Lemma 5.5.3. Suppose that for every college q in Q, the group preference relation $\succeq'_q$ is minimally responsive to $\succeq_q$. Let μ be the matching produced by Algorithm 5.3 and let μ' be a matching such that $\mu' \succeq \mu$. Then $\mu \succeq \mu'$.

Proof. Since μ' is a matching that satisfies the capacity constraints, we can construct an SMIW matching $\mu'_0 \colon P' \to Q' + q'_{00}$ such that for all $1 \leq i \leq |P|$ and $0 \leq j \leq |Q|$, we have $\mu'(p_i) = q_j$ if and only if $\mu_0(p'_i) = q'_{jk}$ for some k.

Since $\mu' \succeq \mu$, we have $\mu'(p_i) \succeq_{p_i} \mu(p_i)$ for every $1 \leq i \leq |P|$ and $\mu'(q_j) \succeq'_{q_j} \mu(q_j)$ for every $1 \leq j \leq |Q|$. Thus $\mu'_0(p'_i) \succeq_{p'_i} \mu_0(p'_i)$ for every $1 \leq i \leq |P|$ and

$$\sum_{1 \leq k \leq c_{q_j}} \psi_{q'_{jk}}(\mu'_0(q'_{jk})) \geq \sum_{1 \leq k \leq c_{q_j}} \psi_{q'_{jk}}(\mu_0(q'_{jk}))$$

for every $1 \leq j \leq |Q|$. Hence, by Lemma 5.4.4, we have $\mu_0(p'_i) \succeq_{p'_i} \mu'_0(p'_i)$ for every $1 \leq i \leq |P|$ and

$$\sum_{1 \leq j \leq |Q|} \sum_{1 \leq k \leq c_{q_j}} \psi_{q'_{jk}}(\mu'_0(q'_{jk})) = \sum_{1 \leq j \leq |Q|} \sum_{1 \leq k \leq c_{q_j}} \psi_{q'_{jk}}(\mu_0(q'_{jk})).$$

Therefore, we have $\mu(p_i) \succeq_{p_i} \mu'(p_i)$ for every $1 \leq i \leq |P|$ and

$$\sum_{1 \leq k \leq c_{q_j}} \psi_{q'_{jk}}(\mu'_0(q'_{jk})) = \sum_{1 \leq k \leq c_{q_j}} \psi_{q'_{jk}}(\mu_0(q'_{jk}))$$

for every $1 \leq j \leq |Q|$. We conclude that $\mu(q_j) \succeq'_{q_j} \mu'(q_j)$ for every $1 \leq j \leq |Q|$. Thus $\mu \succeq \mu'$. $\square$

Theorem 5.5.4. Suppose that for every college q in Q, the group preference relation $\succeq'_q$ is minimally responsive to $\succeq_q$. Algorithm 5.3 is a strategyproof Pareto-stable mechanism for the college admissions problem with weak preferences (for any fixed choice of utility assignment).

Proof. We have shown in Lemma 5.5.2 that Algorithm 5.3 produces a weakly stable matching. Moreover, Lemma 5.5.3 shows that the weakly stable matching produced is not Pareto-dominated by any other matching. Hence, Algorithm 5.3 produces a Pareto-stable matching.

To show that Algorithm 5.3 provides a strategyproof mechanism, suppose student p_i expresses $\succeq'_{p_i}$ instead of their true preference relation $\succeq_{p_i}$, where $1 \leq i \leq |P|$. Let μ and μ' be the matchings produced by Algorithm 5.3 given $\succeq_{p_i}$ and $\succeq'_{p_i}$, respectively. Let μ_0 and μ'_0 be the SMIW matching produced by the call to Algorithm 5.2 (line 8 of Algorithm 5.3) given $\succeq_{p_i}$ and $\succeq'_{p_i}$, respectively.

Notice that in Algorithm 5.3, the only part of the stable marriage instance that depends on the preferences of student p_i is the preference relation corresponding to man p'_i. Since Algorithm 5.2 is strategyproof by Theorem 5.4.7, we have $\mu_0(p'_i) \succeq_{p'_i} \mu'_0(p'_i)$ where $\succeq_{p'_i}$ is the preference relation of man p'_i in the algorithm given $\succeq_{p_i}$. Hence, $\mu(p_i) \succeq_{p_i} \mu'(p_i)$. $\square$

Our algorithm admits an $O(n^4)$-time implementation, where n is the sum of the number of students and the total capacities of all the colleges, because the reduction from CAW to IUAP takes $O(n^2)$ time, and lines 16 and 17 of Algorithm 5.2 can be implemented in $O(n^4)$ time using the version of the incremental Hungarian method discussed in Sections 5.2.3 and 5.3.2.

5.5.2 Further Discussion

In our SMIW and CAW algorithms, we transform the preference relations of the women and colleges into real-valued utility functions. One way to do this is to take

$$\psi_q(p) = |\{p' \in P + \varnothing \colon p \succeq_q p'\}| - |\{p' \in P + \varnothing \colon \varnothing \succeq_q p'\}|.$$

This is by no means the only way. In fact, different ways of assigning the utilities can affect the outcome. Nonetheless, our mechanisms remain strategyproof for the men as long as the utility assignment is fixed and independent of the preferences of the men, as shown in Theorems 5.4.7 and 5.5.4.

We can also consider the scenario where each college expresses their preferences directly in terms of a utility function instead of a preference relation. Such utility functions provide another way to extend preferences over individuals to group preferences. If a college q expresses the utility function ψ_q

over individual students in $P + \varnothing$, we can define the *group preference induced by additive utility* ψ_q as a binary relation $\succeq'_q$ over 2^P such that $S \succeq'_q S'$ if and only if

$$\sum_{p \in S} \psi_q(p) \geq \sum_{p \in S'} \psi_q(p).$$

Our algorithm can accept such utility functions as input in lieu of constructing them by some utility assignment method. It is not hard to see that the mechanism remains Pareto-stable and strategyproof when the group preferences of the colleges are induced by additive utilities.

Chapter 6

Establishing Group Strategyproofness

In this chapter, we establish that the strategyproof (for the men) and Pareto-stable mechanism of Chapter 5 is also group strategyproof (for the men). We do so by showing that the mechanism of Chapter 5 coincides with the group strategyproof Pareto-stable mechanism (for the same model) introduced by Domaniç et al. [17]. [1]

In Section 6.1, we briefly describe the mechanism of [17], and we state the theorem summarizing the group strategyproofness result of [17] and a lemma that is used in Section 6.3; the details and the proofs are available

[1]We wish to clarify the relationship between the results presented in this chapter and [17]. There are two main results established in [17]: a group strategyproof Pareto-stable mechanism for SMIW; the equivalence of that mechanism and the mechanism of this dissertation (Chapter 5). The first result is expected to be included in the dissertation work of coauthor Chi-Kit Lam. The second result is part of this dissertation. In order to present the second result, we will need to review some definitions, a theorem, and a lemma associated with the first result.

in [17]. Then, in Section 6.2, we restate the strategyproof SMIW mechanism of Chapter 5 (Algorithm 5.2) using the notation of [17]. Finally, in Section 6.3, we show that the mechanism of Chapter 5, which is restated in Section 6.2, coincides with the group strategyproof Pareto-stable mechanism of [17], and hence is group strategyproof (Theorem 6.3.17).

6.1 A Group Strategyproof Pareto-Stable Mechanism

The assignment game of Shapley and Shubik [55] involves a two-sided matching market with monetary transfers where the agents have unit-slope linear utility functions. This model has been generalized to allow agents to have continuous, invertible, and increasing utility functions [12, 14, 46]. Some models that generalize both the assignment game and the stable marriage have also been developed, but those models are not concerned with the strategic behavior of agents [27, 60]. The mechanism of [17] casts the stable marriage problem as an appropriate market in the model of Demange and Gale [14]. In Section 6.1.1, we review key concepts in the work of Demange and Gale, and introduce the *tiered-slope market* as a special form of the generalized assignment game in which the slopes of the utility functions are powers of a large fixed number. Then, in Section 6.1.2, we formally define group strategyproofness for the stable marriage model with indifferences. Finally, in Section 6.1.3, we describe our approach for converting a stable marriage market with indifferences into an associated tiered-slope market. Theorem 6.1.1 shows that group strategyproofness for the men in the stable marriage market with indifferences is achieved by man-optimality in the associated tiered-slope market.

6.1.1 Tiered-Slope Markets

The generalized assignment game studied by Demange and Gale [14] involves two disjoint sets I and J of agents, which we call the *men* and the *women*, respectively. We assume that the sets I and J do not contain the element $\varnothing$, which we use to denote being unmatched. For each man i in I and woman j in J, the compensation function $f_{i,j}(u_i)$ represents the compensation that i needs to receive in order to attain utility u_i when he is matched to j. Similarly, for each man i and woman j, the compensation function $g_{i,j}(v_j)$ represents the compensation that j needs to receive in order to attain utility v_j when she is matched to i. Moreover, each man i has a reserve utility r_i and each woman j has a reserve utility s_j.

Throughout this chapter, we assume that the compensation functions are of the form

$$f_{i,j}(u_i) = u_i\lambda^{-a_{i,j}} \qquad \text{and} \qquad g_{i,j}(v_j) = v_j - (b_{i,j}N + \pi_i)$$

and the reserve utilities are of the form

$$r_i = \pi_i\lambda^{a_{i,\varnothing}} \qquad \text{and} \qquad s_j = b_{\varnothing,j}N,$$

such that the following conditions hold: $\pi \in \mathbb{Z}^I$; $N \in \mathbb{Z}$; $\lambda \in \mathbb{Z}$; $a \in \mathbb{Z}^{I \times (J + \varnothing)}$; $b \in \mathbb{Z}^{(I+\varnothing)\times J}$;

$$N > \max_{i \in I} \pi_i \geq \min_{i \in I} \pi_i \geq 1;$$

$$\lambda \geq \max_{(i,j) \in (I+\varnothing)\times J} (b_{i,j} + 1)N \geq \min_{(i,j) \in (I+\varnothing)\times J} (b_{i,j} + 1)N \geq N.$$

We refer to this market as the *tiered-slope market* $\mathcal{M} = (I, J, \pi, N, \lambda, a, b)$. For better readability, we write $\exp_\lambda(\xi)$ to denote λ^ξ.

Let $\mathcal{M}$ be a tiered-slope market $(I, J, \pi, N, \lambda, a, b)$. A *matching* μ is a function from I to $J + \varnothing$ such that for any woman j in J, we have $\mu(i) = j$ for at most one man i. Given a matching μ and a woman j in J, we denote

$$\mu(j) = \begin{cases} i & \text{if } \mu(i) = j \\ 0 & \text{if there is no man } i \text{ in } I \text{ such that } \mu(i) = j. \end{cases}$$

A *utility vector of the men I* is a vector u in $\mathbb{R}^I$. A *utility vector of the women J* is a vector v in $\mathbb{R}^J$. An *outcome* is a triple (μ, u, v), where μ is a matching, u is a utility vector of the men I, and v is a utility vector of the women J. An outcome (μ, u, v) is *feasible* if the following conditions hold for each man i and woman j: if $\mu(i) = j$, then $f_{i,j}(u_i) + g_{i,j}(v_j) \leq 0$; if $\mu(i) = \varnothing$, then $u_i = r_i$; if $\mu(j) = \varnothing$, then $v_j = s_j$. A feasible outcome (μ, u, v) is *individually rational* if $u_i \geq r_i$ and $v_j \geq s_j$ for each man i and woman j. An individually rational outcome (μ, u, v) is *stable* if $f_{i,j}(u_i) + g_{i,j}(v_j) \geq 0$ for each man i and woman j.

A stable outcome (μ, u, v) is *man-optimal* if for any stable outcome

(μ', u', v') we have $u_i \geq u'_i$ for every man $i \in I$. It has been shown that man-optimal outcomes always exist [14, Property 2].

6.1.2 Stable Marriage and Group Strategyproofness

When we formally introduced the SMIW model in Section 5.4, we used P and Q to denote the sets of men and women, respectively, and we used p_i and q_j to denote an individual man and a woman, respectively, where i and j were used to denote the indices of men and woman. In the remainder of this chapter, we adopt the notation of [17] and use I, J, i, and j to denote a set of men, a set of women, an individual man, and a woman, respectively. All the definitions of Section 5.4 carry over here with the revised notation. For the sake of completeness, we enumerate here the other resulting notational changes: the binary relation over $J + \varnothing$ specifying the preference relation of a man i is denoted by $\succeq_i$; the binary relation over $I + \varnothing$ specifying the preference relation of a woman j is denoted by $\succeq_j$; an SMIW instance is a tuple $(I, J, (\succeq_i)_{i \in I}, (\succeq_j)_{j \in J})$, which we call a *stable marriage market*; a *mechanism* is an algorithm that, given a stable marriage market $(I, J, (\succeq_i)_{i \in I}, (\succeq_j)_{j \in J})$, produces a matching μ. In the remainder of this chapter, we do not need the dummy men or women that we introduced in Section 5.4 for convenience. Thus we relax the assumption that $|I| = |J|$ in a stable marriage market $(I, J, (\succeq_i)_{i \in I}, (\succeq_j)_{j \in J})$.

We now formally define group strategyproofness. A mechanism is said to be *group strategyproof (for the men)* if for any two different preference profiles $(\succeq_i)_{i\in I}$ and $(\succeq'_i)_{i\in I}$, there exists a man $i_0 \in I$ with preference relation $\succeq_{i_0}$ different from $\succeq'_{i_0}$ such that $\mu(i_0) \succeq_{i_0} \mu'(i_0)$, where μ and μ' are the matchings produced by the mechanism given $(I, J, (\succeq_i)_{i\in I}, (\succeq_j)_{j\in J})$ and $(I, J, (\succeq'_i)_{i\in I}, (\succeq_j)_{j\in J})$, respectively. (Such a man i_0 belongs to the coalition, but is not matched to a strictly preferred woman by expressing preference relation $\succeq'_{i_0}$ instead of his true preference relation $\succeq_{i_0}$.)

6.1.3 An Associated Tiered-Slope Market and A Mechanism

The group strategyproof Pareto-stable mechanism of [17] for SMIW first creates a tiered-slope market assoicated with the given stable marriage market, in a manner we describe shortly, and then returns a matching corresponding to a man-optimal outcome of this tiered-slope market. The result is summarized in Theorem 6.1.1 below; the proof can be found in [17]. In the remainder of this section, we describe how to create a tiered-slope market assoicated with a stable marriage market, and we present a lemma to be used in Section 6.3 to provide a lower bound on the utilities of men in the man-optimal outcomes of certain markets.

We construct a *tiered-slope market* $\mathcal{M} = (I, J, \pi, N, \lambda, a, b)$ *associated with a stable marriage market* $(I, J, (\succeq_i)_{i\in I}, (\succeq_j)_{j\in J})$ as follows. We take N at least $|I| + 1$, and for each man i in I, we associate with i a fixed and distinct priority π_i from the set $\{1, 2, \ldots, |I|\}$. We convert the preference relations $(\succeq_i)_{i\in I}$ of the men to integer-valued (non-transferable) utilities $a \in \mathbb{Z}^{I\times(J+\varnothing)}$ such that for each man i in I and two elements j_1 and j_2 in $J+\varnothing$, we have $j_1 \succeq_i j_2$ if and only if $a_{i,j_1} \geq a_{i,j_2}$. Similarly, we convert the preference relations $(\succeq_j)_{j\in J}$ of the women to integer-valued (non-transferable) utilities $b \in \mathbb{Z}^{(I+\varnothing)\times J}$ such that for each woman j in J and two elements i_1 and i_2 in $I+\varnothing$, we have $i_1 \succeq_j i_2$ if and only if $b_{i_1,j} \geq b_{i_2,j} \geq 1$. Finally, we take

$$\lambda = \max_{\substack{i\in I+\varnothing \\ j\in J}} (b_{i,j}+1)N.$$

In order to achieve group strategyproofness, we require the parameter N and the priorities π of the men to be independent of the preferences $(\succeq_i)_{i\in I}$ of the men. We further require that b does not depend on the preferences $(\succeq_i)_{i\in I}$ of the men, and that a_{i_0,j_0} does not depend on the other preferences $(\succeq_i)_{i\in I\setminus\{i_0\}}$ for any man $i_0 \in I$ and woman $j_0 \in J+\varnothing$. In other words, each man i_0 is only able to manipulate his own utilities $(a_{i_0,j})_{j\in J+\varnothing}$. One way to satisfy these conditions is as follows: for each man i_0 in I and each element j_0 in $J+\varnothing$, set a_{i_0,j_0} to the number of elements j in $J+\varnothing$ such that $j_0 \succeq_{i_0} j$; for each woman j_0 in J and each element i_0 in $I+\varnothing$, set b_{i_0,j_0} to the number of elements i in $I+\varnothing$ such that $i_0 \succeq_{j_0} i$.

Theorem 6.1.1. If a mechanism produces matchings that correspond to man-optimal outcomes of the tiered-slope markets associated with the stable marriage markets, then it is group strategyproof and Pareto-stable.

The following lemma is used to establish Lemma 6.3.8 in Section 6.3, which provides a lower bound on the utilities of men in the man-optimal outcomes of certain markets; its proof can be found in [17].

Lemma 6.1.2. Let $\mathcal{M} = (I, J, \pi, N, \lambda, a, b)$ be the tiered-slope market associated with stable marriage market $(I, J, (\succeq_i)_{i \in I}, (\succeq_j)_{j \in J})$, and let r denote the reserve utility vector of the men in $\mathcal{M}$. Let r' be a reserve utility vector of the men such that the following conditions hold: (i) $r' \geq r$; (ii) for any man i and any integer k, either $r'_i \exp_\lambda(k)$ is an integer or $0 < r'_i \exp_\lambda(k) < 1$. Let $\mathcal{M}'$ be the market that is equal to $\mathcal{M}$ except that the reserve utilities of the men are given by r', and let (μ, u, v) be a man-optimal outcome in $\mathcal{M}'$. Then $\exp_\lambda(a_{i,\mu(i)}) \leq u_i$ for every man $i \in I$.

6.2 Algorithm 5.2 Revisited

In this section, in order to suit the presentation of the current chapter, we present the strategyproof SMIW mechanism of Chapter 5 (Algorithm 5.2) using the notation of Section 6.1; the mechanism with the new notation is given in Algorithm 6.1. For each woman j, we construct an item, denoted *item*(j), in line 3. For each man i, we construct a dummy item, denoted *item*$_\varnothing(i)$, in line 7, and a multibidder, denoted *multibidder*(i), in line 8, by examining the tiers of preference of the men and the utilities of the women. The set $\{z_i \mid i \in I\}$ of priorities of the multibidders is equal to $\{1, \ldots, |I|\}$, and we assume that the men have no control over the assignment of the priorities. These multibidders and items form an IUAP, from which we obtain a UAP and a greedy MWM M. Finally, in line 13, we use M to determine the match of each man in the solution to the stable marriage instance. It is easy to see that Algorithm 5.2 and Algorithm 6.1 are equivalent, except that Algorithm 6.1 does not require $|I|$ to be equal to $|J|$, as mentioned in Section 6.1.2.

Algorithm 6.1 Algorithm 5.2 revisited using the notation of Section 6.1.

```
Input: A stable marriage market $(I, J, (\succeq_i)_{i \in I}, (\succeq_j)_{j \in J})$.
Output: A Pareto-stable matching $\mu$.
1: for all $j \in J$ do
2:     Convert the preference relation $\succeq_j$ of woman $j$ into utility function
       $\psi_j \colon I + \varnothing \to \mathbb{R}$ that satisfies the following conditions: $\psi_j(\varnothing) = 0$; for any
       $i$ and $i'$ in $I + \varnothing$, we have $i \succeq_j i'$ if and only if $\psi_j(i) \geq \psi_j(i')$. This utility
       assignment should not depend on the preferences of the men.
3:     Construct an item, denoted $item(j)$, corresponding to woman $j$.
4: end for
5: for all $i \in I$ do
6:     Partition the set $J + \varnothing$ into tiers $\tau_i(1), \ldots, \tau_i(K_i)$ according to the
       preference relation of man $i$, such that for any $j$ in $\tau_i(k)$ and $j'$ in $\tau_i(k')$,
       we have $j \succeq_i j'$ if and only if $k \leq k'$.
7:     Construct a dummy item, denoted $item_\varnothing(i)$, corresponding to man $i$.
8:     Construct a multibidder $(\sigma_i, z_i)$, denoted $multibidder(i)$, corresponding
       to man $i$. The priority $z_i$ is uniquely chosen from the set $\{1, \ldots, |I|\}$. The
       sequence $\sigma_i$ has $K_i$ bidders such that for each bidder $\sigma_i(k)$, we define
       $items(\sigma_i(k))$ as $\{item(j) \mid j \in \tau_i(k)\}$ and $w(\sigma_i(k), item(j))$ as $\psi_j(i)$, where
       $item(\varnothing)$ denotes $item_\varnothing(i)$, and $\psi_\varnothing(i)$ denotes 0.
9: end for
10: $B = (T, V) = (\{multibidder(i) \mid i \in I\}, \{item(j) \mid j \in J\} \cup$
    $\{item_\varnothing(i) \mid i \in I\})$.
11: $A = uap(B)$.
12: Compute a greedy MWM $M$ of UAP $A$.
13: return the matching $\mu$ such that for each man $i$ in $I$ and each woman $j$
    in $J$, we have $\mu(i) = j$ if and only if $\sigma_i(k)$ is matched to item $item(j)$ in
    $M$ for some $k$.
```

6.3 Equivalence of the Two Mechanisms

In this section, we fix a stable marriage market $(I, J, (\succeq_i)_{i\in I}, (\succeq_j)_{j\in J})$ and an associated tiered-slope market $\mathcal{M} = (I, J, \pi, N, \lambda, a, b)$. As in Section 6.1, $f_{i,j}(u_i)$ denotes the compensation that man i needs to receive in order to attain utility u_i in $\mathcal{M}$ when he is matched to woman j, and $g_{i,j}(v_j)$ denotes the compensation that woman j needs to receive in order to attain utility v_j in $\mathcal{M}$ when she is matched to man i. We let r denote the reserve utility vector of the men in $\mathcal{M}$, and we let s denote the reserve utility vector of the women in $\mathcal{M}$. We consider an execution of Algorithm 6.1 on the stable marriage market $(I, J, (\succeq_i)_{i\in I}, (\succeq_j)_{j\in J})$, and we let $B = (T, V)$ denote the IUAP constructed at line 10 of this execution. We assume that B is constructed in such a way that the following conditions hold: for each man i, the priority of $\mathit{multibidder}(i)$ is π_i; the offers of the multibidders in B, i.e., the weights of the edges of B, satisfy the conditions stated in the last paragraph of Section 6.3.1 below. With the assumption that these conditions hold, we show in Theorem 6.3.16 that the set of greedy MWMs of $\mathit{uap}(B)$ corresponds to the set of man-optimal matchings of $\mathcal{M}$.

Algorithm 6.1 computes a greedy MWM of the UAP $\mathit{uap}(B)$ in lines 11 and 12. Recall that we defined $\mathit{uap}(B)$ in Section 5.3.1 by giving an algorithm that converts an IUAP to a UAP, namely Algorithm 5.1. In this section, we analyze the executions of Algorithm 5.1 with input B in order to relate the greedy MWMs of $\mathit{uap}(B)$ that Algorithm 6.1 computes to the man-optimal matchings of $\mathcal{M}$. As outlined in Section 1.2.3, our approach is based on the technique of analyzing market instances in which the agents and their utility functions are fixed, while the reserve utilities vary. As noted by Roth and Sotomayor [51, Chapter 9], lowering the reserve utility of an agent is analogous to extending the preferences of an agent in the stable marriage model, a technique used to study structural properties of the stable marriage model. Building on this idea, for each iteration of Algorithm 5.1, we inductively show a bijection (Lemmas 6.3.11 and 6.3.14) from the set of greedy MWMs of the UAP maintained at that iteration to the man-optimal matchings of the corresponding tiered-slope market, where the reserve utilities are adjusted to "reveal" only the preferences that are present in the UAP.

In the preceding sections, the terms "feasible", "individually rational", "stable", and "man-optimal" are used only for outcomes. Throughout this section, however, we also use these terms for payoffs and matchings, as in [14]. Here we briefly review the related definitions. A pair (u, v) consisting of a utility vector u of the men and a utility vector v of the women is a *payoff*. For any feasible outcome (μ, u, v) of a market $\mathcal{M}'$, we say that (u, v) *is a feasible payoff of* $\mathcal{M}'$, and that μ *is compatible with* (u, v). A feasible payoff (u, v) is *individually rational* if $u_i \geq r_i$ for each man i and $v_j \geq s_j$ for each woman j. If an outcome (μ, u, v) is stable (resp., man-optimal) for a market $\mathcal{M}'$, then we say that μ *is a stable* (resp., *man-optimal*) *matching of* $\mathcal{M}'$, and that (u, v) *is a stable* (resp., *man-optimal*) *payoff in* $\mathcal{M}'$. For a given market, there is a unique man-optimal payoff, but there can be more than one man-optimal matching.

6.3.1 Edge Weights of the IUAP

We start our discussion by introducing some useful mappings from items to women and from man-woman pairs to bidders. Recall that, for each woman j, $\mathit{item}(j)$ is constructed at line 3 of Algorithm 6.1, and for each man i, $\mathit{item}_{\varnothing}(i)$ is constructed at line 7 and $\mathit{multibidder}(i)$ is constructed at line 8. For any non-dummy item v in V, we define $\mathit{woman}(v)$ as the woman in J associated with v. (Thus for each woman j, $\mathit{woman}(\mathit{item}(j))$ is equal to j.) For any dummy item v in V, we define $\mathit{woman}(v)$ as $\varnothing$. (Thus for each man i, $\mathit{woman}(\mathit{item}_{\varnothing}(i))$ is equal to $\varnothing$.) For any subset V' of V, we define $\mathit{women}(V')$ as $\{\mathit{woman}(v) \mid v \in V'\}$. For any man i in I and any element j in $J + \varnothing$, we define $\mathit{bidder}(i, j)$ as the bidder in $\mathit{multibidder}(i)$ such that $\mathit{women}(\mathit{items}(\mathit{bidder}(i, j)))$ contains j. Remark: For each man i, the dummy item $\mathit{item}_{\varnothing}(i)$ belongs to $\mathit{items}(\mathit{bidder}(i, \varnothing))$.

In the remainder of the paper, for any man i and any woman j, we use the shorthand $w(i, j)$ to denote $w(\mathit{bidder}(i, j), \mathit{item}(j))$, and $w(i, \varnothing)$ to denote $w(\mathit{bidder}(i, \varnothing), \mathit{item}_{\varnothing}(i))$.

For any matching μ of $\mathcal{M}$, we define $b(\mu)$ as $\sum_{\mu(j)\neq\varnothing} b_{\mu(j),j} + \sum_{\mu(j)=\varnothing} b_{\varnothing,j}$, and we define $w(\mu)$ as $\sum_{\mu(j)\neq\varnothing} w(\mu(j), j)$.

We assume that the IUAP B is constructed so that the $w(i, j)$'s satisfy the following conditions: (i) for any man i, $w(i, \varnothing) = 0$; (ii) for any two matchings μ and μ' of $\mathcal{M}$, $w(\mu) \geq w(\mu')$ if and only if $b(\mu) \geq b(\mu')$. It is easy to see that one way to satisfy these conditions is to set $\psi_j(i) = b_{i,j} - b_{\varnothing,j}$ in Algorithm 6.1.

6.3.2 Tiered-Slope Market Matchings and Greedy MWMs

In this section, we first introduce certain mappings from the tiered-slope market matchings to the UAP matchings, and we define weights for these tiered-slope market matchings. Then, in Lemma 6.3.6, we show that these mappings define bijections from certain sets of tiered-slope market matchings — those maximizing the weights we introduce — to the sets of greedy MWMs. We start with some useful definitions.

For any configuration $C = (A, B)$ and any man i, we define the predicate $P_{\mathit{all}}(C, i)$ to hold if i has revealed all of his acceptable tiers in C, i.e., if $\mathit{bidder}(i, \varnothing)$ belongs to A.

Lemma 6.3.1. Let $C = (A, B)$ be a configuration and let i be a man such that $P_{\mathit{all}}(C, i)$ holds. Then for each greedy MWM M of A, some bidder associated with i is matched in M.

Proof. Suppose the claim does not hold, and let M be a greedy MWM of A such that there is no bidder associated with i that is matched in M. Then, since $\mathit{bidder}(i, \varnothing)$ and $\mathit{item}_{\varnothing}(i)$ belong to A, $w(i, \varnothing) = 0$, and $\mathit{priority}(\mathit{bidder}(i, \varnothing)) > 0$, we deduce that the matching $M' = M + (\mathit{bidder}(i, \varnothing), \mathit{item}_{\varnothing}(i))$ is an MWM of A such that $\mathit{priority}(M') > \mathit{priority}(M)$, a contradiction. □

We define C_1 as the configuration (A, B) where A is the UAP that reveals only the first bidder of each multibidder in T. We say that an execution of Algorithm 5.1 invoked with input B is *canonical* if C_1 is equal to the configuration that the program variable C stores at some iteration of this execution. We define C_F as the unique final configuration of any canonical execution, i.e., $(uap(B), B)$. We say that a configuration is *relevant* if it is equal to the configuration that the program variable C stores at iteration t_1 or a subsequent iteration of some canonical execution, where t_1 is the iteration at which $C_1 = C$.

Given a relevant configuration $C = (A, B)$, we now introduce a mapping from certain matchings in the tiered-slope market $\mathcal{M}$ to the matchings in the UAP A, and a weight function for these matchings in $\mathcal{M}$. Let $C = (A, B)$ be a relevant configuration and let μ be a matching of $\mathcal{M}$ such that for each man-woman pair (i, j) matched in μ, $\mathit{bidder}(i, j)$ belongs to A. Then, we define $\Phi_C(\mu)$ as the matching

$$\bigcup_{\mu(i)\neq\varnothing} \{(\mathit{bidder}(i,\mu(i)), \mathit{item}(\mu(i)))\} \cup \bigcup_{\mu(i)=\varnothing \wedge P_{all}(C,i)} \{(\mathit{bidder}(i,\varnothing), \mathit{item}_{\varnothing}(i))\}.$$

It is easy to see that Φ_C is an injection and that $\Phi_C(\mu)$ is a matching of the UAP A. Furthermore, we have $w(\Phi_C(\mu)) = w(\mu)$ since $w(\mathit{bidder}(i,\varnothing), \mathit{item}_{\varnothing}(i)) = 0$ for any man i by condition (i) of Section 6.3.1. For any man i, we define

$$\pi_{\varnothing}(C,\mu,i) = \begin{cases} 0 & \text{if } \mu(i) \neq \varnothing \\ \pi_i & \text{if } \mu(i) = \varnothing \text{ and } P_{all}(C,i) \\ (1-\lambda^{-1}) & \text{if } \mu(i) = \varnothing \text{ and } \neg P_{all}(C,i). \end{cases}$$

We define $\pi_{\varnothing}(C,\mu)$ as $\sum_{i \in I} \pi_{\varnothing}(C,\mu,i)$. Remark: It is easy to see that

$$\pi_{\varnothing}(C,\mu) = \sum_{\mu(i)=\varnothing} \pi_{\varnothing}(C,\mu,i) = \sum_{\mu(i)=\varnothing \wedge P_{all}(C,i)} \pi_i + \sum_{\mu(i)=\varnothing \wedge \neg P_{all}(C,i)} (1-\lambda^{-1}).$$

Finally, we define $W_C(\mu)$ as

$$N \cdot b(\mu) + \sum_{\mu(i)\neq\varnothing} \pi_i + \pi_{\varnothing}(C,\mu).$$

Remark: It is easy to see that

$$W_C(\mu) = N \cdot b(\mu) + \sum_{\mu(i)\neq\varnothing \vee P_{all}(C,i)} \pi_i + \sum_{\mu(i)=\varnothing \wedge \neg P_{all}(C,i)} (1-\lambda^{-1}), \tag{6.1}$$

and that the second term in the RHS is equal to $\mathit{priority}(\Phi_C(\mu))$.

We now show that Φ_C is invertible in certain cases.

Lemma 6.3.2. Let $C = (A, B)$ be a relevant configuration and let M be a matching of A such that the following conditions hold: (1) for each man i, at most one bidder associated with i is matched in M; (2) for each man i, if $P_{all}(C,i)$ holds then a bidder associated with i is matched in M. Then there exists a unique matching μ of $\mathcal{M}$ such that $\Phi_C(\mu)$ is equal to M.

Proof. Let μ denote the matching such that $\mu(i) = j$ if and only if $(\mathit{bidder}(i,j), \mathit{item}(j))$ belongs to M. It is easy to see by condition (1) that μ is a matching of $\mathcal{M}$. Moreover, condition (2) implies that if $\mu(i) = \varnothing$ for a man i, then $(\mathit{bidder}(i,\varnothing), \mathit{item}_{\varnothing}(i))$ belongs to M. The claim follows since Φ_C is an injection and $\Phi_C(\mu)$ is equal to M. □

Lemma 6.3.3. Let $C = (A, B)$ be a relevant configuration and let M be a greedy MWM of A. Then there exists a unique matching μ of $\mathcal{M}$ such that $\Phi_C(\mu)$ is equal to M.

Proof. It is sufficient to prove that M satisfies the two conditions of Lemma 6.3.2. Condition (1) is satisfied because Lemmas 5.3.1 and 5.3.2 imply that, for each man i, M matches at most one bidder associated with i. Lemma 6.3.1 implies that Condition (2) is satisfied. $\square$

We now show that, given a relevant configuration $C = (A, B)$, Φ_C is a bijection from a certain set of matchings of $\mathcal{M}$ to the set of greedy MWMs of A. We start with some useful lemmas and definitions that help us to define this set.

Lemma 6.3.4. Let $C = (A, B)$ be a relevant configuration and let (i, j) be a man-woman pair such that $\mathit{bidder}(i, j)$ belongs to A. Then $a_{i,j} \geq a_{i,\varnothing}$.

Proof. If the least preferred acceptable tier of a man i is revealed at some iteration, i.e., if $\mathit{bidder}(i, \varnothing)$ is added at line 4 of Algorithm 5.1, then for any configuration C' that results in a subsequent iteration, $P_{\mathit{all}}(C', i)$ holds, and by Lemma 6.3.1, there is no bidder associated with i in $\mathit{ready}(C')$. Hence no other tier of i is subsequently revealed. $\square$

For any relevant configuration C and any man i, we define $\mathit{least}(C, i)$ as the nonempty subset of $J+\varnothing$ in the least preferred tier of i that is revealed in C, i.e., $\mathit{women}(\mathit{items}(\mathit{bidder}(t, |\mathit{bidders}(C, t)|)))$, where t denotes $\mathit{multibidder}(i)$.

Remark: $a_{i,j} = a_{i,j'} \geq a_{i,\varnothing}$ for any j and j' belonging to $\mathit{least}(C, i)$, where the inequality follows from Lemma 6.3.4 and is tight if and only if $P_{\mathit{all}}(C, i)$ holds.

For any relevant configuration C and any matching μ of $\mathcal{M}$, we define the predicate $P_{\mathit{least}}(C, \mu)$ to hold if for each man-woman pair (i, j) matched in μ, the woman j belongs to $\mathit{least}(C, i)$. Remark: For any relevant configuration $C = (A, B)$ and any matching μ such that $P_{\mathit{least}}(C, \mu)$ holds, it is easy to see that $\Phi_C(\mu)$ is well-defined because for each man-woman pair (i, j) matched in μ, $\mathit{bidder}(i, j)$ belongs to A.

The following lemma is only used to prove Lemma 6.3.6.

Lemma 6.3.5. Let $C = (A, B)$ be a relevant configuration, and let μ_0 and μ be two matchings of $\mathcal{M}$ such that $P_{\mathit{least}}(C, \mu_0)$ holds, $\Phi_C(\mu)$ is a greedy MWM of A, and $w(\mu_0) < w(\mu)$. Then $W_C(\mu_0) < W_C(\mu)$.

Proof. Let M_0 denote $\Phi_C(\mu_0)$ and let M denote $\Phi_C(\mu)$. Since M is a greedy MWM of A, Lemma 5.3.2 implies that $P_{\mathit{least}}(C, \mu)$ holds. The symmetric difference of M_0 and M, denoted $M_0 \oplus M$, corresponds to a collection $\mathcal{S}$ of vertex-disjoint paths and cycles. Let $\langle Q_1, \ldots, Q_{|\mathcal{S}|}\rangle$ be an arbitrary permutation of $\mathcal{S}$. For any integer k such that $1 \leq k \leq |\mathcal{S}|$, let X_k denote the edges of Q_k that belong to M, let X'_k denote the edges of Q_k that belong to M_0, and let M_k denote the matching $(M_{k-1} \setminus X'_k) \cup X_k$. Remark: It is easy to see that $M_{|\mathcal{S}|}$ is equal to M.

We start by showing that, for each integer k such that $1 \leq k \leq |\mathcal{S}|$, M_k satisfies the two conditions of Lemma 6.3.2. It is easy to see that condition (1) holds because $P_{\mathit{least}}(C, \mu_0)$ and $P_{\mathit{least}}(C, \mu)$ imply that, for any integer k such that $0 \leq k \leq |\mathcal{S}|$, any bidder that is matched in M_k is the least preferred bidder of the associated man that is revealed in A. We now address condition (2). Since $P_{least}(C, \mu_0)$ and $P_{least}(C, \mu)$ hold, the definitions of M_0 and M imply that for each man i such that $P_{all}(C, i)$ holds, both M_0 and M match the bidder $bidder(i, \varnothing)$. It follows that, for each man i such that $P_{all}(C, i)$ holds, $bidder(i, \varnothing)$ is not an endpoint of any path in $\mathcal{S}$, and thus $bidder(i, \varnothing)$ is matched in M_k for all $1 \leq k \leq |\mathcal{S}|$, establishing condition (2). Having established that M_k satisfies the two conditions of Lemma 6.3.2, for each integer k such that $1 \leq k \leq |\mathcal{S}|$, we define μ_k as the matching of $\mathcal{M}$ such that $\Phi_C(\mu_k)$ is equal to M_k. We now establish two simple but useful claims.

Claim 1: $w(M_k) > w(M_{k-1})$ for at least one k such that $1 \leq k \leq |\mathcal{S}|$. The claim follows directly from the fact that $w(M_0) = w(\mu_0) < w(\mu) = w(M) = w(M_{|\mathcal{S}|})$.

Claim 2: $w(M_k) \geq w(M_{k-1})$ for all $1 \leq k \leq |\mathcal{S}|$. For the sake of contradiction, suppose that the claim fails for some integer k. Then $(M \setminus X_k) \cup X'_k$ is a matching of A with weight higher than that of M, a contradiction since M is an MWM of A.

Having established these two claims, we now complete the proof of the lemma by showing that the following two conditions hold for any $1 \leq k \leq |\mathcal{S}|$: (a) if $w(M_k) > w(M_{k-1})$ then $W_C(\mu_k) > W_C(\mu_{k-1})$; and (b) if $w(M_k) = w(M_{k-1})$ then $W_C(\mu_k) \geq W_C(\mu_{k-1})$.

We first address condition (a). Let k be an integer such that $1 \leq k \leq |\mathcal{S}|$ and $w(M_k) > w(M_{k-1})$. Our goal is to establish that $W_C(\mu_k) >$

$W_C(\mu_{k-1})$. Since $|I| - |\Phi_C(\mu')|$ is equal to $|\{i \mid \mu'(i) = \varnothing \wedge \neg P_{all}(C,i)\}|$ for any μ', equality (6.1) and the associated remark imply that the difference $W_C(\mu_k) - W_C(\mu_{k-1})$ is equal to

$$N \cdot (b(\mu_k) - b(\mu_{k-1})) + (\mathit{priority}(M_k) - \mathit{priority}(M_{k-1})) + \\ (|M_{k-1}| - |M_k|)(1 - \lambda^{-1}). \tag{6.2}$$

Since $w(\mu_k) = w(M_k) > w(M_{k-1}) = w(\mu_{k-1})$, condition (ii) stated in Section 6.3.1 implies that $b(\mu_k) > b(\mu_{k-1})$. Then, since N, $b(\mu_k)$, $b(\mu_{k-1})$, and the priorities are integers, and since $N > \max_{i \in I} \pi_i$, we deduce that the first term of (6.2) is at least $1 + \max_{i \in I} \pi_i$. Thus, in order to establish that $W_C(\mu_k) > W_C(\mu_{k-1})$, it is enough to show that the sum of the second and third term of (6.2) is greater than $-1 - \max_{i \in I} \pi_i$. If Q_k is a cycle, then it is easy to see that $\mathit{matched}(M_k) = \mathit{matched}(M_{k-1})$, and hence that both the second and third terms of (6.2) are zero. In the remainder of this paragraph, we address the case where Q_k is a path. In this case, it is easy to see that $\mathit{matched}(M_{k-1}) \setminus \mathit{matched}(M_k)$ contains at most one bidder. Then, since $\min_{i \in I} \pi_i > 0$, we deduce that the second term of (6.2) is at least $-\max_{i \in I} \pi_i$. Finally, since $-1 \leq |M_{k-1}| - |M_k| \leq 1$, we conclude that the third term of (6.2) is greater than -1, as required.

We now address condition (b). Let k be an integer such that $1 \leq k \leq |\mathcal{S}|$ and $w(M_k) = w(M_{k-1})$. Our goal is to establish that $W_C(\mu_k) \geq W_C(\mu_{k-1})$. Again, the difference $W_C(\mu_k) - W_C(\mu_{k-1})$ is equal to (6.2). In this case, since $w(\mu_k) = w(M_k) = w(M_{k-1}) = w(\mu_{k-1})$, condition (ii) stated in Section 6.3.1

implies that $b(\mu_k) = b(\mu_{k-1})$, and hence that the first term of (6.2) is zero. Thus, in order to establish that $W_C(\mu_k) \geq W_C(\mu_{k-1})$, it remains to show that the sum of the second and third term of (6.2) is nonnegative. If Q_k is a cycle, then it is easy to see that $\mathit{matched}(M_k) = \mathit{matched}(M_{k-1})$, and hence that both the second and third terms of (6.2) are zero. In the remainder of this paragraph, we address the case where Q_k is a path. In this case, it is easy to see that $\mathit{matched}(M_k) \neq \mathit{matched}(M_{k-1})$. Since (as argued in the second paragraph of the proof) any bidder that is matched in M_k or in M_{k-1} is the least preferred bidder of the associated man that is revealed in A, we deduce that $\mathit{priority}(M_k) \neq \mathit{priority}(M_{k-1})$. We conclude that $\mathit{priority}(M_k) > \mathit{priority}(M_{k-1})$, for otherwise $(M \setminus X_k) \cup X'_k$ is an MWM of A with priority higher than that of M, a contradiction since M is a greedy MWM of A. It follows that the second term of (6.2) is at least 1 since the priorities are integers. Finally, since $-1 \leq |M_{k-1}| - |M_k| \leq 1$, we conclude that the third term of (6.2) is greater than -1, as required. $\square$

Lemma 6.3.6. Let $C = (A, B)$ be a relevant configuration and let X be the set of all matchings μ of $\mathcal{M}$ such that $P_{\mathit{least}}(C, \mu)$ holds. Let X^* denote the set $\{\mu \mid \mu \in X \wedge W_C(\mu) = \max_{\mu' \in X} W_C(\mu')\}$. Then, Φ_C is a bijection from X^* to the set of greedy MWMs of A.

Proof. Let M be a greedy MWM of A and let μ be the matching (by Lemma 6.3.3) of $\mathcal{M}$ such that $\Phi_C(\mu)$ is equal to M. Lemma 5.3.2 implies that $P_{\mathit{least}}(C, \mu)$ holds; thus μ belongs to X. Let μ^* be a matching in X^* and let M^* denote $\Phi_C(\mu^*)$. Note that M^* is a matching of A. Since Φ_C is an injection, it remains to show that $W_C(\mu) = \max_{\mu' \in X} W_C(\mu')$ and that M^* is a greedy MWM of A.

Since $W_C(\mu) \leq \max_{\mu' \in X} W_C(\mu') = W_C(\mu^*)$, Lemma 6.3.5 implies that $w(\mu) \leq w(\mu^*)$. Then, since M is an MWM of A, we deduce that M^* is an MWM of A. Since M is a greedy MWM, and hence an MCMWM of A, we deduce that $|M^*| \leq |M|$. We consider two cases.

Case 1: $|M^*| < |M|$. Let x be a bidder in $\mathit{matched}(M) \setminus \mathit{matched}(M^*)$ such that there exists an MWM, call it M', of A having $\mathit{matched}(M') = \mathit{matched}(M^*) + x$; the exchange property of $\mathit{matroid}(A)$ implies the existence of such a bidder x and matching M'. Let i' denote the man associated with x. We first argue that M' satisfies the two conditions of Lemma 6.3.2. By definition, the men corresponding to the bidders in $\mathit{matched}(M^*)$ are distinct; let I' denote the set of men corresponding to these bidders, i.e., $I' = \{i \mid \mu^*(i) \neq \varnothing \vee P_{\mathit{all}}(C, i)\}$. Since $P_{\mathit{least}}(C, \mu^*)$ holds, we deduce that, for each man i in I', the bidder in $\mathit{matched}(M^*)$ associated with i corresponds to the least preferred tier of i that is revealed in C. Lemma 5.3.2 implies that x corresponds to the least preferred tier of i' that is revealed in C. Then, since x does not belong to $\mathit{matched}(M^*)$, the results of the preceding two sentences imply that i' does not belong to I', and thus that M' satisfies condition (1). We now address condition (2). Matching M^* satisfies condition (2) by definition. Thus M' satisfies condition (2) since any bidder matched by M^* is also matched by M'. Since M' satisfies conditions (1) and (2), Lemma 6.3.2 implies that there is a matching, call it μ', of $\mathcal{M}$ such that $\Phi_C(\mu')$ is equal to M'. Since both M' and M^* are MWMs of A, condition (ii) stated in Section 6.3.1 implies that $b(\mu') = b(\mu^*)$. Since $\mathit{matched}(M')$ properly contains $\mathit{matched}(M^*)$, we deduce that the set $\{i \mid \mu'(i) \neq \varnothing \vee P_{\mathit{all}}(C, i)\}$ properly contains the set $\{i \mid \mu^*(i) \neq \varnothing \vee P_{\mathit{all}}(C, i)\}$. Then, since $\pi_i > (1 - \lambda^{-1})$ for each man i, the results of the preceding two sentences imply that $W_C(\mu') > W_C(\mu^*)$, contradicting the definition of μ^*.

Case 2: $|M^*| = |M|$. Since both M and M^* are MWMs of A, condition (ii) stated in Section 6.3.1 implies that $b(\mu) = b(\mu^*)$. Since $|\mathit{matched}(M)| = |\mathit{matched}(M^*)|$, we deduce that the cardinality of $\{i \mid \mu(i) \neq \varnothing \vee P_{\mathit{all}}(C, i)\}$ is equal to the cardinality of $\{i \mid \mu^*(i) \neq \varnothing \vee P_{\mathit{all}}(C, i)\}$. Then, by using the results of the preceding two sentences and the remark regarding the second term in the RHS of (6.1), we deduce that $W_C(\mu) - W_C(\mu^*) = \mathit{priority}(\Phi_C(\mu)) - \mathit{priority}(\Phi_C(\mu^*))$. The latter expression is nonnegative since M is a greedy MWM of A, and it is nonpositive since $W_C(\mu^*) = \max_{\mu' \in X} W_C(\mu')$. Thus we deduce that $W_C(\mu) = \max_{\mu' \in X} W_C(\mu')$ and that M^* is a greedy MWM of A. $\square$

6.3.3 Revealing Preferences in the Tiered-Slope Market

Recall that Algorithm 5.1 iteratively reveals the bidders, which correspond to the tiers of men, and the state of the revealed bidders are captured in a configuration. In this section, we first show how to adjust the reserve utilities of the men to obtain markets identical to the tiered-slope market except that only the tiers that are revealed in a configuration are acceptable (Lemma 6.3.7).

Then, we inductively show a bijection (Lemma 6.3.11 and 6.3.14) from the set of greedy MWMs of the UAP maintained at each iteration of Algorithm 5.1 to the man-optimal matchings of the corresponding market with adjusted reserve utilities. Finally, we establish our results in Theorems 6.3.16 and 6.3.17.

For any relevant configuration C, we define *reserve*(C) as the reserve utility vector r' of I such that for each man i,

$$r'_i = \max\left\{r_i, (1-\lambda^{-1})\exp_\lambda(a_{i,j})\right\} = \begin{cases} \pi_i \exp_\lambda(a_{i,\varnothing}) & \text{if } P_{\mathit{all}}(C,i) \\ (1-\lambda^{-1})\exp_\lambda(a_{i,j}) & \text{otherwise,} \end{cases}$$

where j is some element in *least*(C,i).

For any reserve utility vector r' of I such that $r' \geq r$, we define $\mathcal{M}(r')$ as the market that is equal to $\mathcal{M}$ except that the reserve utilities of the men are given by r'. For any relevant configuration C, we define $\mathcal{M}(C)$ as $\mathcal{M}(\mathit{reserve}(C))$. Lemma 6.3.7 below shows that, for any relevant configuration C, only the tiers of men that are revealed in C are "acceptable" in $\mathcal{M}(C)$.

Lemma 6.3.7. Let $C = (A, B)$ be a relevant configuration, let (μ, u, v) be an individually rational outcome for $\mathcal{M}(C)$, and let (i, j) be a man-woman pair matched in μ. Then $\mathit{bidder}(i, j)$ belongs to A.

Proof. We have

$$\begin{aligned} f_{i,j}(u_i) &\le -g_{i,j}(v_j) = -v_j + Nb_{i,j} + \pi_i \\ &\le N\left(b_{i,j} - b_{\varnothing,j}\right) + \pi_i \\ &\le Nb_{i,j} - N + \pi_i \\ &\le \lambda - 2N + \pi_i \\ &\le \lambda - 2, \end{aligned}$$

where the inequalities are justified as follows: the first inequality follows from the feasibility of (μ, u, v); the second inequality follows from the individual rationality of (μ, u, v), which implies $v_j \ge s_j = Nb_{\varnothing,j}$; the third inequality follows since $b_{\varnothing,j} \ge 1$; the fourth inequality follows since $\lambda \ge \max_{i,j}(b_{i,j}+1)N$; the fifth inequality follows since $N > \max_{i\in I} \pi_i$. Then, since $f_{i,j}(u_i) \le \lambda - 2$, we deduce that $u_i \le (\lambda - 2)\exp_\lambda(a_{i,j})$.

Let r' denote *reserve*(C) and let j' be an arbitrary element in *least*(C, i). Assume the claim of the lemma is false: thus $a_{i,j} < a_{i,j'}$. Then, since $u_i \le (\lambda - 2)\exp_\lambda(a_{i,j})$, we conclude that $u_i < (1 - \lambda^{-1})\exp_\lambda(a_{i,j'})$, and hence that $u_i < r'_i$, contradicting individual rationality. □

The following lemma provides a lower bound on the utilities of men in the man-optimal outcomes. It is used in the proof of Lemma 6.3.14.

Lemma 6.3.8. Let C be a relevant configuration, let $(\mu, \overline{u}, \underline{v})$ be a man-optimal outcome for $\mathcal{M}(C)$, and let (i, j) be a man-woman pair matched in μ. Then $\overline{u}_i \ge \exp_\lambda(a_{i,j})$.

Proof. Let r' denote $\mathit{reserve}(C)$. We show that r' satisfies the conditions required by Lemma 6.1.2, then the claim follows from that lemma. By definition, r' is at least r, so it satisfies condition (i). For any man i, if $P_{\mathit{all}}(C, i)$ holds, then r'_i is equal to $\pi_i \exp_\lambda(a_{i,\varnothing})$, and it is easy to see that $r'_i \exp_\lambda(k)$ is an integer for any integer $k \geq -a_{i,\varnothing}$ and that $0 < r'_i \exp_\lambda(k) < 1$ for any integer $k < -a_{i,\varnothing}$; otherwise, r'_i is equal to $(1-\lambda^{-1})\exp_\lambda(a_{i,j})$ where j is some element in $\mathit{least}(C, i)$, and it is easy to see that $r'_i \exp_\lambda(k)$ is an integer for any integer $k > -a_{i,j}$ and that $0 < r'_i \exp_\lambda(k) < 1$ for any integer $k \leq -a_{i,j}$. Thus, r' satisfies condition (ii) as well. □

For a matching μ satisfying $P_{\mathit{least}}(C, \mu)$, the following lemma gives a necessary and sufficient condition for μ to be a stable matching of $\mathcal{M}(C)$. The proof of the lemma is quite involved and is deferred to Section 6.3.4.

Lemma 6.3.9. Let C be a relevant configuration and let X be the set of all matchings μ of $\mathcal{M}(C)$ such that $P_{\mathit{least}}(C, \mu)$ holds. Assume that there exists at least one stable outcome, denoted (μ, u, v), for $\mathcal{M}(C)$ such that μ belongs to X. Then, a matching μ^* that belongs to X is compatible with the stable payoff (u, v) if and only if $W_C(\mu^*) = \max_{\mu' \in X} W_C(\mu')$.

For any relevant configuration C, we define the predicate $P_{\mathit{opt}}(C)$ to hold if for each man-optimal matching μ of $\mathcal{M}(C)$, $P_{\mathit{least}}(C, \mu)$ holds. Remark: It is easy to see that $P_{\mathit{least}}(C_1, \mu)$ holds for any matching μ of $\mathcal{M}(C_1)$, and thus $P_{\mathit{opt}}(C_1)$ holds. For a relevant configuration C satisfying $P_{\mathit{opt}}(C)$, the following lemma characterizes the man-optimal matchings of $\mathcal{M}(C)$.

Lemma 6.3.10. Let C be a relevant configuration such that $P_{opt}(C)$ holds. Let X be the set of all matchings μ of $\mathcal{M}(C)$ such that $P_{least}(C, \mu)$ holds. Then, a matching μ is a man-optimal matching of $\mathcal{M}(C)$ if and only if μ belongs to X and $W_C(\mu) = \max_{\mu' \in X} W_C(\mu')$.

Proof. Let X^* denote the set $\{\mu \mid \mu \in X \wedge W_C(\mu) = \max_{\mu' \in X} W_C(\mu')\}$. Since $P_{opt}(C)$ implies that all man-optimal matchings of $\mathcal{M}(C)$ are included in X, and since there is at least one man-optimal matching [14, Property 2], we deduce that X contains a man-optimal, and hence stable, matching. Thus, Lemma 6.3.9 implies that the set of stable matchings of $\mathcal{M}(C)$ is equal to X^*, and that each matching in X^* is compatible with the man-optimal payoff in $\mathcal{M}(C)$. □

Lemma 6.3.11. Let $C = (A, B)$ be a relevant configuration such that $P_{opt}(C)$ holds. Then, Φ_C is a bijection from the set of man-optimal matchings of $\mathcal{M}(C)$ to the set of greedy MWMs of A.

Proof. Let X be the set of all matchings $\mathcal{M}(C)$ of $\mathcal{M}$ such that $P_{least}(C, \mu)$ holds. Let X^* denote the set $\{\mu \mid \mu \in X \wedge W_C(\mu) = \max_{\mu' \in X} W_C(\mu')\}$. Lemma 6.3.10 implies that the set of man-optimal matchings of $\mathcal{M}(C)$ is equal to X^*. Then the claim follows from Lemma 6.3.6. □

Having established the correspondence between the man-optimal matchings and the greedy MWMs given a relevant configuration C such that $P_{opt}(C)$ holds, we now show inductively in Lemma 6.3.14 that $P_{opt}(C)$ holds for all relevant configurations C. We start with two lemmas that are useful in proving

Lemma 6.3.14; the second one (Lemma 6.3.13) is also used in the proof of Theorem 6.3.16.

Lemma 6.3.12. Let C be a relevant configuration such that $P_{opt}(C)$ holds, let r' denote *reserve*(C), and let $(\overline{u}, \underline{v})$ denote the man-optimal payoff in $\mathcal{M}(C)$. Then, for each bidder in *ready*(C), we have $\overline{u}_i = r'_i$, where i denotes the man associated with that bidder.

Proof. Let C be (A, B). If a bidder belongs to *ready*(C), then it is not matched in any greedy MWM of A, and the definition of Φ_C and Lemma 6.3.11 imply that it is not matched in any man-optimal matching of $\mathcal{M}(C)$. Then the claim of the lemma follows from the stability of $(\overline{u}, \underline{v})$. □

Lemma 6.3.13. Let r' and r'' be two reserve utility vectors of the men such that $r' \geq r'' \geq r$. Let (u', v') and (u'', v'') be the man-optimal payoffs of $\mathcal{M}(r')$ and $\mathcal{M}(r'')$, respectively. Then $u' \geq u''$.

Proof. The claim of the lemma follows directly from [14, Property 3]. □

Lemma 6.3.14. Let C be a relevant configuration such that for each relevant configuration C' that precedes C, $P_{opt}(C')$ holds. Then $P_{opt}(C)$ holds.

Proof. Consider an arbitrary canonical execution that produces C at some iteration and let (A, B) be the configuration C. For the sake of contradiction, assume $P_{opt}(C)$ does not hold. Let $(\mu, \overline{u}, \underline{v})$ be a man-optimal outcome for $\mathcal{M}(C)$ such that $P_{least}(C, \mu)$ does not hold and let (i, j) be a man-woman pair matched in μ such that j does not belong to *least*(C, i). Let j' be an arbitrary element of $\mathit{least}(C, i)$. Since Lemma 6.3.7 implies that $\mathit{bidder}(i, j)$ belongs to A, we deduce that $a_{i,j} > a_{i,j'}$. Let C' be the configuration at the beginning of the iteration that reveals the tier of i corresponding to $\mathit{least}(C, i)$, i.e., the iteration at which $\mathit{bidder}(i, j')$ is added at line 4 of Algorithm 5.1. Let j'' be an arbitrary element of $\mathit{least}(C', i)$, let r' denote $\mathit{reserve}(C')$, and let $(\overline{u}', \underline{v}')$ denote the man-optimal payoff in $\mathcal{M}(C')$. Then, we deduce that $a_{i,j} \geq a_{i,j''} > a_{i,j'} \geq a_{i,\varnothing}$, where the last inequality follows from Lemma 6.3.4. Thus, $P_{\mathit{all}}(C', i)$ does not hold, and hence $r'_i = (1-\lambda^{-1}) \exp_\lambda(a_{i,j''})$. Since $\mathit{bidder}(i, j')$ belongs to $\mathit{ready}(C')$, and since $P_{\mathit{opt}}(C')$ holds, Lemma 6.3.12 implies that $\overline{u}'_i = r'_i = (1 - \lambda^{-1}) \exp_\lambda(a_{i,j''})$. On the other hand, since the man-woman pair (i, j) is matched in μ, which is a man-optimal matching of $\mathcal{M}(C)$, Lemma 6.3.8 implies that $\overline{u}_i \geq \exp_\lambda(a_{i,j})$. Combining the results of the preceding two sentences, we conclude that $\overline{u}_i \geq \exp_\lambda(a_{i,j}) > (1 - \lambda^{-1}) \exp_\lambda(a_{i,j''}) = \overline{u}'_i$, contradicting Lemma 6.3.13 since $\mathit{reserve}(C) \leq \mathit{reserve}(C')$. □

We are now ready to establish our equivalence result in Theorem 6.3.16 and the group strategyproofness result in Theorem 6.3.17. We start with a useful lemma.

Lemma 6.3.15. For each greedy MWM M of $\mathit{uap}(B)$ and each man i, some bidder associated with i is matched in M.

Proof. Suppose the claim does not hold, and let M be a greedy MWM of $\mathit{uap}(B)$ and let i be a man such that no bidder associated with i is matched in M. Then all of the bidders associated with i belong to $\mathit{uap}(B)$, for otherwise $\mathit{ready}(C_F)$ is nonempty, contradicting the definition of C_F (recall that

$C_F = (\mathit{uap}(B), B)$ is the unique final configuration of any canonical execution). Thus, we deduce that $P_{\mathit{all}}(C_F, i)$ holds, and Lemma 6.3.1 implies that some bidder associated with i is matched in M, a contradiction. □

Theorem 6.3.16. Φ_{C_F} is a bijection from the set of man-optimal matchings of $\mathcal{M}$ to the set of greedy MWMs of $\mathit{uap}(B)$.

Proof. Observe that $P_{\mathit{opt}}(C_F)$ holds by repeated application of Lemma 6.3.14. Let $(\overline{u}, \underline{v})$ denote the man-optimal payoff in $\mathcal{M}(C_F)$. Let r' denote $\mathit{reserve}(C_F)$ and recall that r is the reserve utility vector of the men in $\mathcal{M}$. We now prove two useful claims.

Claim 1: Any man-optimal outcome $(\mu, \overline{u}, \underline{v})$ for $\mathcal{M}(C_F)$ is stable for $\mathcal{M}$. Let $(\mu, \overline{u}, \underline{v})$ be a man-optimal outcome for $\mathcal{M}(C_F)$. Since $\mathcal{M}(C_F)$ and $\mathcal{M}$ differ only in the reserve utility vectors of the men, it is enough to show that for each man i who is unmatched in μ, we have $\overline{u}_i = r_i$. Let i be a man who is unmatched in μ. Observe that $P_{\mathit{all}}(C_F, i)$ holds, for otherwise no bidder associated with i is matched in $\Phi_{C_F}(\mu)$, which is a greedy MWM of $\mathit{uap}(B)$ by Lemma 6.3.11, contradicting Lemma 6.3.15. Thus $r'_i = \pi_i \exp_\lambda(a_{i,\varnothing}) = r_i$. Then, by the stability of $(\mu, \overline{u}, \underline{v})$ for $\mathcal{M}(C_F)$, we conclude that $\overline{u}_i = r'_i = r_i$.

Claim 2: Any man-optimal outcome for $\mathcal{M}$ is stable for $\mathcal{M}(C_F)$. Claim 1 and Lemma 6.3.13 imply that the payoff $(\overline{u}, \underline{v})$, which is the man-optimal payoff in $\mathcal{M}(C_F)$, is also the man-optimal payoff in $\mathcal{M}$. Let $(\mu, \overline{u}, \underline{v})$ be a man-optimal outcome for $\mathcal{M}$. As in the proof of Claim 1, since $\mathcal{M}(C_F)$ and $\mathcal{M}$ differ only in the reserve utility vectors of the men, it is enough to show that for each man i who is unmatched in μ, we have $\overline{u}_i = r'_i$. Let i be a man who is unmatched in μ. By the stability of $(\mu, \overline{u}, \underline{v})$ for $\mathcal{M}$, we deduce that $\overline{u}_i = r_i$. By the individual rationality of $(\overline{u}, \underline{v})$ for $\mathcal{M}(C_F)$, we deduce that $\overline{u}_i \geq r'_i$. Since $r' \geq r$, we conclude that $\overline{u}_i = r_i = r'_i$.

Claims 1 and 2, and Lemma 6.3.13 imply that the set of man-optimal matchings of $\mathcal{M}$ is equal to the set of man-optimal matchings of $\mathcal{M}(C_F)$. Then the theorem follows from Lemma 6.3.11 since $P_{opt}(C_F)$ holds. $\square$

Theorem 6.3.17. The SMIW mechanism of Section 5.4 is group strategyproof.

Proof. As we mentioned in Section 6.2, Algorithm 6.1 implements the mechanism of Section 5.4. Given an instance of the stable marriage market with indifferences, if we construct a tiered-slope market $\mathcal{M}$ associated with this instance and we run Algorithm 6.1 by setting the edge weights as described in Section 6.3.1, Theorem 6.3.16 implies that the output corresponds to a man-optimal matching of $\mathcal{M}$. Then the result follows from Theorem 6.1.1. $\square$

6.3.4 Proof of Lemma 6.3.9

The purpose of this section is to prove Lemma 6.3.9. We start with some useful lemmas.

Lemma 6.3.18. Let C be a relevant configuration, let μ be a matching of $\mathcal{M}(C)$, and let v be a utility vector of the women such that $v_j = s_j$ for each woman j who is unmatched in μ. Then,

$$\sum_{j\in J} v_j - W_C(\mu) + \pi_\varnothing(C,\mu) = \sum_{\mu(j)\neq\varnothing} g_{\mu(j),j}(v_j).$$

Proof.

$$\begin{aligned}
\sum_{j\in J} v_j - W_C(\mu) + \pi_\varnothing(C,\mu) &= \sum_{j\in J} v_j - N\cdot b(\mu) - \sum_{\mu(i)\neq\varnothing} \pi_i \\
&= \sum_{\mu(j)\neq\varnothing} v_j - N\sum_{\mu(j)\neq\varnothing} b_{\mu(j),j} - \sum_{\mu(j)\neq\varnothing} \pi_{\mu(j)} \\
&\quad + \sum_{\mu(j)=\varnothing} s_j - N\sum_{\mu(j)=\varnothing} b_{\varnothing,j} \\
&= \sum_{\mu(j)\neq\varnothing} \big(v_j - b_{\mu(j),j}N - \pi_{\mu(j)}\big) \\
&= \sum_{\mu(j)\neq\varnothing} g_{\mu(j),j}(v_j),
\end{aligned}$$

where the third equality follows since $s_j = Nb_{\varnothing,j}$. □

Lemma 6.3.19. Let C be a relevant configuration and let (μ,u,v) be a stable outcome for $\mathcal{M}(C)$. Then,

$$W_C(\mu) = \sum_{\mu(i)\neq\varnothing} f_{i,\mu(i)}(u_i) + \sum_{j\in J} v_j + \pi_\varnothing(C,\mu).$$

Proof. The stability and feasibility of (μ,u,v) imply that $f_{i,j}(u_i) + g_{i,j}(v_j) = 0$ for each man-woman pair (i,j) matched in μ. Thus, $\sum_{\mu(i)\neq\varnothing} f_{i,\mu(i)}(u_i) + \sum_{\mu(j)\neq\varnothing} g_{\mu(j),j}(v_j) = 0$, and the claim follows from Lemma 6.3.18, since the stability of (μ, u, v) implies that $v_j = s_j$ for each woman j who is unmatched in μ. $\square$

Lemma 6.3.20. Let C be a relevant configuration, let μ and μ' be two matchings such that $P_{least}(C, \mu)$ and $P_{least}(C, \mu')$ hold, let r' denote *reserve*(C), and let u be a utility vector such that for each man i, $u_i = r'_i$ if $\mu(i) = \varnothing$ or $\mu'(i) = \varnothing$. Then $\sum_{\mu(i) \neq \varnothing} f_{i,\mu(i)}(u_i) + \pi_{\varnothing}(C, \mu) = \sum_{\mu'(i) \neq \varnothing} f_{i,\mu'(i)}(u_i) + \pi_{\varnothing}(C, \mu')$.

Proof. Let $f_{i,\varnothing}(u_i)$ denote 0. We show that $f_{i,\mu(i)}(u_i) + \pi_{\varnothing}(C, \mu, i) = f_{i,\mu'(i)}(u_i) + \pi_{\varnothing}(C, \mu', i)$ for each man i. Let i be an arbitrary man. We consider six cases.

Case 1: $\mu(i) \neq \varnothing$ and $\mu'(i) \neq \varnothing$. Then $\pi_{\varnothing}(C, \mu, i) = \pi_{\varnothing}(C, \mu', i) = 0$. $P_{least}(C, \mu)$ and $P_{least}(C, \mu')$ and imply that $a_{i,\mu(i)} = a_{i,\mu'(i)}$, and hence $f_{i,\mu(i)}(u_i) = f_{i,\mu'(i)}(u_i)$.

Case 2: $\mu(i) = \mu'(i) = \varnothing$. In this case $\pi_{\varnothing}(C, \mu, i)$ (resp., $\pi_{\varnothing}(C, \mu', i)$) is independent of μ (resp., μ').

Case 3 (resp., case 4): $\mu(i) \neq \varnothing$ and $\mu'(i) = \varnothing$ (resp., $\mu(i) = \varnothing$ and $\mu'(i) \neq \varnothing$) and $P_{all}(C, i)$. Since $P_{all}(C, i)$ holds, we deduce that $r'_i = \pi_i \exp_{\lambda}(a_{i,\varnothing})$. Then, since $u_i = r'_i$, $P_{least}(C, \mu)$ (resp., $P_{least}(C, \mu')$) implies that $f_{i,\mu(i)}(u_i) + \pi_{\varnothing}(C, \mu, i) = f_{i,\mu(i)}(r'_i) = \pi_i$ (resp., $f_{i,\mu'(i)}(u_i) + \pi_{\varnothing}(C, \mu', i) = f_{i,\mu'(i)}(r'_i) = \pi_i$). Since $\mu'(i) = \varnothing$ (resp., $\mu(i) = \varnothing$), we deduce that $f_{i,\mu'(i)}(u_i) + \pi_{\varnothing}(C, \mu', i)$ (resp., $f_{i,\mu(i)}(u_i) + \pi_{\varnothing}(C, \mu, i)$) is equal to $0 + \pi_i = \pi_i$.

Case 5 (resp., case 6): $\mu(i) \neq \varnothing$ and $\mu'(i) = \varnothing$ (resp., $\mu(i) = \varnothing$ and $\mu'(i) \neq \varnothing$) and $\neg P_{all}(C, i)$. Let j denote $\mu(i)$ (resp., $\mu'(i)$). Since $P_{least}(C, \mu)$ (resp., $P_{least}(C, \mu')$) and $\neg P_{all}(C, i)$, we deduce that $r'_i = (1 - \lambda^{-1}) \exp_{\lambda}(a_{i,j})$. Then, since $u_i = r'_i$, we deduce that $f_{i,\mu(i)}(u_i) + \pi_{\varnothing}(C, \mu, i) = f_{i,\mu(i)}(r'_i) =$

$(1-\lambda^{-1})$ (resp., $f_{i,\mu'(i)}(u_i)+\pi_{\varnothing}(C,\mu',i) = f_{i,\mu'(i)}(r'_i) = (1-\lambda^{-1})$). Since $\mu'(i)=\varnothing$ (resp., $\mu(i)=\varnothing$), we deduce that $f_{i,\mu'(i)}(u_i)+\pi_{\varnothing}(C,\mu',i)$ (resp., $f_{i,\mu(i)}(u_i)+\pi_{\varnothing}(C,\mu,i)$) is equal to $0+(1-\lambda^{-1})=(1-\lambda^{-1})$. □

Lemma 6.3.21. Let C be a relevant configuration and let r' denote *reserve*(C). Let X be the set of all matchings μ of $\mathcal{M}(C)$ such that $P_{least}(C,\mu)$ holds. Let (μ,u,v) be a stable outcome for $\mathcal{M}(C)$ such that μ belongs to X. Let μ^* be a matching in X such that $W_C(\mu^*)=\max_{\mu'\in X} W_C(\mu')$. Then the following claims hold: (1) $u_i = r'_i$ if $\mu^*(i)=\varnothing$; (2) $v_j = s_j$ if $\mu^*(j)=\varnothing$; (3) $\sum_{\mu(i)\neq\varnothing} f_{i,\mu(i)}(u_i)+\pi_{\varnothing}(C,\mu) = \sum_{\mu^*(i)\neq\varnothing} f_{i,\mu^*(i)}(u_i)+\pi_{\varnothing}(C,\mu^*)$.

Proof. Let $\mathcal{S}$ denote the symmetric difference of μ and μ^*. It is easy to see that $\mathcal{S}$ is a collection of positive length paths and cycles. In order to prove Claim (1) of the lemma, consider an arbitrary man i such that $\mu^*(i)=\varnothing$. If $\mu(i)=\varnothing$, then the stability of (u,v) establishes the claim, so assume that $\mu(i)\neq\varnothing$. Then, i is an endpoint of a path in $\mathcal{S}$; let P denote this path. The edges of P alternate between edges that are matched in μ and edges that are matched in μ^*. We consider two cases.

Case 1: The other endpoint of P is a man i' such that $\mu(i')=\varnothing$ and $\mu^*(i)\neq\varnothing$. Let P be $\langle i=i_1, j_1, \ldots, j_k, i_{k+1}=i'\rangle$ for some $k\geq 1$. Then, since $\mu(i_\ell)=j_\ell$ for $1\leq \ell\leq k$, the stability of (μ,u,v) implies that

$$\sum_{1\leq \ell\leq k} [f_{i_\ell,j_\ell}(u_{i_\ell})+g_{i_\ell,j_\ell}(v_{j_\ell})]=0.$$

The stability of (μ, u, v) also implies that

$$\sum_{1 \le \ell \le k} \left[f_{i_{\ell+1}, j_\ell}(u_{i_{\ell+1}}) + g_{i_{\ell+1}, j_\ell}(v_{j_\ell}) \right] \ge 0.$$

By subtracting the latter equation from the former, we obtain the following, since $P_{\mathit{least}}(C, \mu)$ and $P_{\mathit{least}}(C, \mu^*)$ imply $f_{i_\ell, j_\ell}(u_{i_\ell}) = f_{i_\ell, j_{\ell-1}}(u_{i_\ell})$ for $1 < \ell \le k$:

$$\begin{aligned} 0 &\ge f_{i,j_1}(u_i) - f_{i',j_k}(u_{i'}) + \sum_{1 \le \ell \le k} \left[g_{i_\ell, j_\ell}(v_{j_\ell}) - g_{i_{\ell+1}, j_\ell}(v_{j_\ell}) \right] \\ &= (f_{i,j_1}(u_i) - \pi_i) - (f_{i',j_k}(u_{i'}) - \pi_{i'}) + N \sum_{1 \le \ell \le k} \left(b_{i_{\ell+1}, j_\ell} - b_{i_\ell, j_\ell} \right). \end{aligned} \tag{6.3}$$

Observe that

$$f_{i,j_1}(u_i) - \pi_i \ge f_{i,j_1}(r'_i) - \pi_i \ge (1 - \lambda^{-1}) - \pi_i > -N, \tag{6.4}$$

since the individual rationality of (u, v) implies $u_i \ge r'_i$ and since $P_{\mathit{least}}(C, \mu)$ holds. Also observe that

$$f_{i',j_k}(u_{i'}) - \pi_{i'} = f_{i',j_k}(r'_{i'}) - \pi_{i'} \le 0,$$

since the stability of (μ, u, v) implies $u_{i'} = r'_{i'}$ and since $P_{\mathit{least}}(C, \mu^*)$ holds. These two observations imply that the third term in (6.3) is nonpositive, for otherwise it would be at least N (since all b values are integers), violating the inequality. The third term in (6.3) is nonnegative, for otherwise μ^* could be augmented along P to yield another matching μ' such that $P_{\mathit{least}}(C, \mu')$ and

$W_C(\mu') - W_C(\mu^*) \geq N - \pi_{i'} > 0$ (since all b values are integers and $N > \pi_{i'}$), contradicting the definition of μ^*. Thus, we may rewrite inequality (6.3) as

$$f_{i',j_k}(r'_{i'}) - \pi_{i'} \geq f_{i,j_1}(u_i) - \pi_i. \tag{6.5}$$

We consider the following four subcases.

Case 1.1: $P_{all}(C, i)$ and $P_{all}(C, i')$. Since $P_{least}(C, \mu^*)$ and $P_{all}(C, i')$ hold, we deduce that $r'_{i'} = \pi_{i'} \exp_\lambda(a_{i',j_k})$, and hence that LHS of (6.5) is 0. Since $P_{least}(C, \mu)$ and $P_{all}(C, i)$ hold, we deduce that $r'_i = \pi_i \exp_\lambda(a_{i',j_1})$, and hence that RHS of (6.5) is at least 0 by the first inequality of (6.4). Thus, we conclude that $f_{i,j_1}(u_i) = \pi_i$, which implies $u_i = r'_i$.

Case 1.2: $\neg P_{all}(C, i)$ and $\neg P_{all}(C, i')$. First observe that $\pi_{i'} > \pi_i$, for otherwise μ^* could be augmented along P to yield another matching μ' such that $P_{least}(C, \mu')$ and $W_C(\mu') - W_C(\mu^*) = \pi_i - \pi_{i'} > 0$, contradicting the definition of μ^*. Since $P_{least}(C, \mu^*)$ holds and $P_{all}(C, i')$ does not hold, we deduce that $r'_{i'} = (1 - \lambda^{-1}) \exp_\lambda(a_{i',j_k})$, and hence that LHS of (6.5) is $(1 - \lambda^{-1}) - \pi_{i'}$. However, the RHS of (6.5) is at least $(1 - \lambda^{-1}) - \pi_i$ by (6.4), contradicting the inequality $\pi_{i'} > \pi_i$.

Case 1.3: $\neg P_{all}(C, i)$ and $P_{all}(C, i')$. This case is not possible, for otherwise μ^* could be augmented along P to yield another matching μ' such that $P_{least}(C, \mu')$ and $W_C(\mu') - W_C(\mu^*) = \pi_i - (1 - \lambda^{-1}) > 0$ (since $\pi_i > (1 - \lambda^{-1})$ for each man i), contradicting the definition of μ^*.

Case 1.4: $P_{all}(C, i)$ and $\neg P_{all}(C, i')$. Since $P_{least}(C, \mu^*)$ holds and $P_{all}(C, i')$ does not hold, we deduce that $r'_{i'} = (1 - \lambda^{-1}) \exp_\lambda(a_{i',j_k})$, and hence that LHS of (6.5) is $(1-\lambda^{-1}) - \pi_{i'} < 0$. Since $P_{least}(C,\mu)$ and $P_{all}(C,i)$ hold, we deduce that $r'_i = \pi_i \exp_\lambda(a_{i',j_1})$, and hence that RHS of (6.5) is at least 0 by the first inequality of (6.4), a contradiction.

Case 2: The other endpoint of P is a woman j such that $\mu(j) \neq \varnothing$ and $\mu^*(j) = \varnothing$. Let P be $\langle i = i_1, j_1, \ldots, i_k, j_k = j\rangle$ for some $k \geq 1$. Then, since $\mu(i_\ell) = j_\ell$ for $1 \leq \ell \leq k$, the stability of (μ, u, v) implies that

$$\sum_{1\leq \ell \leq k} [f_{i_\ell,j_\ell}(u_{i_\ell}) + g_{i_\ell,j_\ell}(v_{j_\ell})] = 0.$$

The stability of (μ, u, v) also implies that

$$\sum_{1<\ell\leq k} \left[f_{i_\ell,j_{\ell-1}}(u_{i_\ell}) + g_{i_\ell,j_{\ell-1}}(v_{j_{\ell-1}})\right] \geq 0.$$

By subtracting the latter equation from the former, we obtain the following, since $P_{least}(C,\mu)$ and $P_{least}(C,\mu^*)$ imply $f_{i_\ell,j_\ell}(u_{i_\ell}) = f_{i_\ell,j_{\ell-1}}(u_{i_\ell})$ for $1 < \ell \leq k$:

$$\begin{aligned} 0 &\geq f_{i,j_1}(u_i) + \sum_{1\leq\ell\leq k} g_{i_\ell,j_\ell}(v_{j_\ell}) - \sum_{1<\ell\leq k} g_{i_\ell,j_{\ell-1}}(v_{j_{\ell-1}}) \\ &= (f_{i,j_1}(u_i) - \pi_i) + v_j + N\left[\sum_{1<\ell\leq k} b_{i_\ell,j_{\ell-1}} - \sum_{1\leq\ell\leq k} b_{i_\ell,j_\ell}\right] \\ &= (f_{i,j_1}(u_i) - \pi_i) + (v_j - s_j) + N\left[\left(b_{\varnothing,j} + \sum_{1<\ell\leq k} b_{i_\ell,j_{\ell-1}}\right) - \sum_{1\leq\ell\leq k} b_{i_\ell,j_\ell}\right]. \end{aligned} \tag{6.6}$$

Observe that (6.4) holds for the same reasons pointed out in Case 1, and that the second term in (6.6) is nonnegative by the individual rationality of

(u, v). Moreover, the third term is nonnegative, for otherwise μ^* could be augmented along P to yield another matching μ' such that $P_{least}(C, \mu')$ and $W_C(\mu') - W_C(\mu^*) \geq N$ (since all b values are integers), contradicting the definition of μ^*. We consider two subcases.

Case 2.1: $P_{all}(C, i)$. Then $P_{least}(C, \mu)$ implies that $r'_i = \pi_i \exp_\lambda(a_{i',j_1})$, and we deduce that the first term in (6.6) is at least 0 by the first inequality of (6.4). Thus, we conclude that $v_j = s_j$ and that $f_{i,j_1}(u_i) = \pi_i$, which implies $u_i = r'_i$.

Case 2.2: $\neg P_{all}(C, i)$. In this case the third term in (6.6) is positive, and thus is at least N (since all b values are integers), for otherwise (if it is 0), μ^* could be augmented along P to yield another matching μ' such that $P_{least}(C, \mu')$ and $W_C(\mu') - W_C(\mu^*) = \pi_i - (1 - \lambda^{-1}) > 0$, contradicting the definition of μ^*. Since (6.4) implies that the first term of (6.6) is greater than $-N$, we deduce that the sum of the three terms in (6.6) is positive, a contradiction.

In order to prove Claim (2) of the lemma, consider an arbitrary woman j such that $\mu^*(j) = \varnothing$. If $\mu(j) = \varnothing$, then the stability of (u, v) establishes the claim, so assume that $\mu(j) \neq \varnothing$. Then, j is an endpoint of a path in $\mathcal{S}$; let P denote this path. The edges of P alternate between edges that are matched in μ and edges that are matched in μ^*. We consider two cases.

Case 1: The other endpoint of P is a woman j' such that $\mu(j') = \varnothing$ and $\mu^*(j) \neq \varnothing$. Let P be $\langle j = j_1, i_1, \ldots, i_k, j_{k+1} = j' \rangle$ for some $k \geq 1$. Then, since $\mu(i_\ell) = j_\ell$ for $1 \leq \ell \leq k$, the stability of (μ, u, v) implies that

$$\sum_{1 \leq \ell \leq k} [f_{i_\ell, j_\ell}(u_{i_\ell}) + g_{i_\ell, j_\ell}(v_{j_\ell})] = 0.$$

The stability of (μ, u, v) also implies that

$$\sum_{1 \leq \ell \leq k} \left[f_{i_\ell, j_{\ell+1}}(u_{i_\ell}) + g_{i_\ell, j_{\ell+1}}(v_{j_{\ell+1}})\right] \geq 0.$$

By subtracting the latter equation from the former, we obtain the following, since $P_{least}(C, \mu)$ and $P_{least}(C, \mu^*)$ imply $f_{i_\ell, j_\ell}(u_{i_\ell}) = f_{i_\ell, j_{\ell+1}}(u_{i_\ell})$ for $1 \leq \ell \leq k$:

$$\begin{aligned} 0 &\geq \sum_{1 \leq \ell \leq k} \left[g_{i_\ell, j_\ell}(v_{j_\ell}) - g_{i_\ell, j_{\ell+1}}(v_{j_{\ell+1}})\right] \\ &= v_j - v_{j'} + N \sum_{1 \leq \ell \leq k} \left(b_{i_\ell, j_{\ell+1}} - b_{i_\ell, j_\ell}\right) \\ &= (v_j - s_j) + N \left[\left(b_{\varnothing, j} + \sum_{1 \leq \ell \leq k} b_{i_\ell, j_{\ell+1}}\right) - \left(b_{\varnothing, j'} + \sum_{1 \leq \ell \leq k} b_{i_\ell, j_\ell}\right)\right], \qquad (6.7) \end{aligned}$$

where the last equality follows since $\mu(j') = \varnothing$ implies $v_{j'} = s_{j'} = b_{\varnothing, j'} N$. The first term in (6.7) is nonnegative by the individual rationality of (u, v). The second term is nonnegative, for otherwise μ^* could be augmented along P to yield another matching μ' such that $P_{least}(C, \mu')$ and $W_C(\mu') - W_C(\mu^*) \geq N$ (since all b values are integers), contradicting the definition of μ^*. Thus, we conclude that $v_j = s_j$.

Case 2: The other endpoint of P is a man i such that $\mu(i) \neq \varnothing$ and $\mu^*(i) = \varnothing$. This case is identical to case 2 in the proof of Claim (1) above, and hence we conclude that $v_j = s_j$.

We now prove Claim (3) of the lemma. Stability of (μ, u, v) implies $u_i = r'_i$ if $\mu(i) = \varnothing$. Thus Claim (3) follows using Claim (1) and Lemma 6.3.20. $\square$

Proof of Lemma 6.3.9. Let μ^* be a matching in X such that $W_C(\mu^*)$ is equal to $\max_{\mu' \in X} W_C(\mu')$. We show that $W_C(\mu) = W_C(\mu^*)$ and that μ^* is compatible with the stable payoff (u, v). We have

$$\begin{aligned} W_C(\mu^*) \geq W_C(\mu) &= \sum_{\mu(i) \neq \varnothing} f_{i,\mu(i)}(u_i) + \sum_{j \in J} v_j + \pi_{\varnothing}(C, \mu) \\ &= \sum_{\mu^*(i) \neq \varnothing} f_{i,\mu^*(i)}(u_i) + \sum_{j \in J} v_j + \pi_{\varnothing}(C, \mu^*), \qquad (6.8) \end{aligned}$$

where the first equality follows from Lemma 6.3.19, and the second equality follows from Claim (3) of Lemma 6.3.21. Then, by using Claim (2) of Lemma 6.3.21 and Lemma 6.3.18, we may rewrite (6.8) as

$$0 \geq \sum_{\mu^*(i) \neq \varnothing} f_{i,\mu^*(i)}(u_i) + \sum_{\mu^*(j) \neq \varnothing} g_{\mu^*(j),j}(v_j) = \sum_{\mu^*(i) = j \wedge j \neq \varnothing} \left[f_{i,j}(u_i) + g_{i,j}(v_j) \right]. \qquad (6.9)$$

Then, since the stability of (u, v) implies that $f_{i,j}(u_i) + g_{i,j}(v_j) \geq 0$ for each man-woman pair (i, j), we deduce the following: the inequality of (6.9) is tight, and hence that of (6.8) is also tight; $f_{i,j}(u_i) + g_{i,j}(v_j) = 0$ for each man-woman pair (i, j) matched in μ^*. Thus, Claims (1) and (2) of Lemma 6.3.21 imply that μ^* is compatible with the stable payoff (u, v). $\square$

Chapter 7

Concluding Remarks

The main contributions of this dissertation are fast algorithms for implementing the VCG mechanism for certain compactly representable special cases of unit-demand auctions, and a group strategyproof Pareto-stable mechanism for the stable marriage model with incomplete and weak preferences (SMIW). The latter mechanism is the first strategyproof and Pareto-stable mechanism for SMIW. This mechanism is developed using a framework based on two variants of unit-demand auctions, namely UAPs and IUAPs, which allow us to generalize the well-known deferred acceptance algorithm. We have established the group strategyproofness of the mechanism by showing that it coincides with the more recent group strategyproof Pareto-stable mechanism (for the same model) of Domaniç et al. [17].

A number of questions arise naturally given the results of this dissertation and the techniques used to derive these results. In this chapter, we state several questions that are of potential interest for future research.

7.1 Further Variants of UDALEWs

We studied the problem of computing a VCG outcome of a UDALEW in Chapter 2, and we studied two special cases of UDALEWs in Chapter 4. These special cases are motivated by applications to the scheduling domain, and enable us to solve two variants of the problem of scheduling unit jobs with rejection in $O(n \log n)$ time. The (more general) UDALEWs can be used to solve other variants of this problem, for instance the problem of scheduling (with rejection) unit jobs having symmetric earliness and tardiness penalties with respect to a common due date, but the algorithm that we present in Chapter 2 is not as fast as the $O(n \log n)$-time algorithms of Chapter 4. One can investigate further special cases of UDALEWs that allow for algorithms faster than that of Chapter 2. For instance, solving the special case of UDALEWs where the item qualities form the sequence $\langle 1, 1, 2, 2, \ldots, d, d \rangle$ addresses the aforementioned variant of the problem with symmetric earliness and tardiness penalties with respect to a common due date.[1]

One can investigate further generalizations of UDALEWs that allow for efficient algorithms for computing a VCG outcome. One can seek applications

[1]It can be shown that the acceptance order notion of Section 4.2.1 is also well-defined in this special case, so one can hope to follow the footsteps of Section 4.2 to devise an algorithm for this case. Let us see what happens if we try to apply the techniques of Section 4.2.2 to compute the acceptance orders. Consider the example illustrated in Figure 4.1, in which we try to determine whether the job with index 10 precedes $\sigma_9[7]$ in σ_{10}. Imagine that the slots now have "qualities" that form the sequence $\langle 1, 1, 2, 2, \ldots \rangle$; the qualities are not depicted explicitly in the figure because the indices of the slots played the role of qualities in the model of Section 4.2. The approach of comparing the weights of two matchings, as illustrated in Figure 4.1b, fails because only some of the jobs, instead of all, in the set $\mathit{best}(9, 6) \setminus \mathit{better}(\sigma_9[7])$ are shifted to a slot that have one lower "quality".

and algorithms for the models that incorporate one, or a combination, of the following to UDALEWs: a budget constraint with each unit-demand bid; a lower or an upper bound on the qualities of the items that is acceptable for each unit-demand bid; a quality threshold with each unit-demand bid so that the offer is given by an affine function up to the threshold, and is a constant after the threshold; vectors of fixed dimensions to play the role of bid slopes and item qualities (in this case the offer is the intercept plus the dot product of the slope and the quality); the operation of removing a unit-demand bid from the auction.

7.2 Further Applications and Generalizations of IUAPs

In Sections 5.2 and 5.3, we introduced the UAPs and IUAPs, and we presented a framework based on these two variants of unit-demand auctions for generalizing the deferred acceptance algorithm. One can investigate further applications and generalizations of UAPs, IUAPs, and this framework. In Section 5.4.1, we described how we exploit our framework to devise a Pareto-stable and group strategyproof mechanism for SMIW; however, in this application, the IUAP mechanism is invoked in a somewhat unusual manner, in that the offers appearing in the unit-demand bids of the multibidder corresponding to a man i are determined by the women — i only gets to determine the set of women to be included in each unit-demand bid in his bid sequence. A possible direction for further investigation is to seek more direct applications of IUAPs, e.g., to implement mechanisms in a variant of unit-demand auctions that incorporates budget constraints into each unit-demand bid [4, 22]. Another possible direction is to generalize the UAPs and IUAPs further, e.g., to allow each component of a unit-demand bid to specify a piecewise linear utility function, instead of a single valuation, as in the work of Dütting et al. [21].

7.3 Threshold Prices and Group Strategyproofness

We established the strategyproofness of our mechanism by utilizing the notion of threshold prices in IUAPs. However, our proof of group strategyproofness in Chapter 6 did not rely on this notion. Instead, we showed that it coincides with a more recent group strategyproof Pareto-stable mechanism. One can investigate whether it is possible to directly establish group strategyproofness by generalizing the notion of threshold prices to a group of items and by generalizing the related lemma (Lemma 5.3.8) to a group of multibidders. In this regard, a plausible approach is to delay the introduction of any unit-demand bids of a group of multibidders until the members of that group are the sole unmatched multibidders whose sequences of unit-demand bids have not been exhausted. One can then seek to characterize the utilities that the members of the group would obtain from any sequence of unit-demand bids that they might choose to submit.

7.4 Stable Marriage with Indifferences

A number of interesting questions related to the stable marriage model with indifferences arise naturally given the existence of various stability notions and other game-theoretic properties of interest. Here we give some examples of these questions.

Recall that we mentioned three stability notions in Section 1.2 regarding the stable marriage model when indifferences are allowed: weak stability, strong stability, and super-stability. Our mechanism returns a weakly stable matching. One can investigate whether any of the other stability notions is achievable by a Pareto-optimal and (group) strategyproof mechanism. Here we briefly review the notion of strong stability. We say that a man i and a woman j form a *weakly blocking pair* with respect to a matching M if (1) i weakly prefers j to his match in M, (2) j weakly prefers i to her match in M, and (3) either (3a) i prefers j to his match in M or (3b) j prefers i to her match in M. A matching that does not admit a weakly blocking pair is said to be *strongly stable*. Unfortunately, it is not difficult to exhibit SMCW instances for which the set of strongly stable matchings is empty (see, e.g., Manlove [42, Section 3.3.1]). For SMIW instances that admit a strongly stable matching, it is desirable for the mechanism to return a matching that is Pareto-optimal and strongly stable. Does our mechanism achieve this property? If not, can we extend it to do so? Can we provide a (partial) characterization of the class of strategyproof (resp., group strategyproof) mechanisms for computing a Pareto-stable matching? Is every strategyproof Pareto-stable mechanism for

SMIW also group strategyproof?

Bibliography

[1] Canadian resident matching service. URL `http://carms.ca/`.

[2] A. Abdulkadiroğlu, P. A. Pathak, and A. E. Roth. Strategy-proofness versus efficiency in matching with indifferences: redesigning the NYC high school match. *The American Economic Review*, 99:1954–1978, 2009.

[3] A. Aggarwal, A. Barnoy, S. Khuller, D. Kravets, and B. Schieber. Efficient minimum cost matching and transportation using the quadrangle inequality. *Journal of Algorithms*, 19:116–143, 1995.

[4] G. Aggarwal, S. Muthukrishnan, D. Pál, and M. Pál. General auction mechanism for search advertising. In *Proceedings of the 18th World Wide Web Conference*, pages 241–250, April 2009.

[5] I. Ashlagi and P. Shi. Improving community cohesion in school choice via correlated-lottery implementation. *Operations Research*, 62:1247–1264, 2014.

[6] I. Ashlagi and P. Shi. Optimal allocation without money: An engineering approach. *Management Science*, 62:1078–1097, 2016.

[7] R. E. Burkard. Monge properties, discrete convexity and applications. *European Journal of Operational Research*, 176:1–14, 2007.

[8] N. Chen. On computing Pareto stable assignments. In *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science*, pages 384–395, March 2012.

[9] N. Chen and A. Ghosh. Algorithms for Pareto stable assignment. In *Proceedings of the Third International Workshop on Computational Social Choice*, pages 343–354, September 2010.

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

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, Cambridge, MA, 2nd edition, 2001.

[12] V. P. Crawford and E. M. Knoer. Job matching with heterogeneous firms and workers. *Econometrica*, 49:437–450, 1981.

[13] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[14] G. Demange and D. Gale. The strategy structure of two-sided matching markets. *Econometrica*, 53:873–888, 1985.

[15] N. O. Domaniç and C. G. Plaxton. Scheduling unit jobs with a common deadline to minimize the sum of weighted completion times and rejection penalties. In *Proceedings of the 25th International Symposium on Algorithms and Computation*, December 2014. Full version available as UTCS Technical Report TR–14–11.

[16] N. O. Domaniç, C.-K. Lam, and C. G. Plaxton. Bipartite matching with linear edge weights. In *Proceedings of the 27th International Symposium on Algorithms and Computation*, December 2016. Full version available as UTCS Technical Report TR–16–15.

[17] N. O. Domaniç, C.-K. Lam, and C. G. Plaxton. Group strategyproof Pareto-stable marriage with indifferences via the generalized assignment game, July 2017. URL `https://arxiv.org/abs/1707.01496`. To appear in Proceedings of the 10th International Symposium on Algorithmic Game Theory.

[18] N. O. Domaniç, C.-K. Lam, and C. G. Plaxton. Strategyproof Pareto-stable mechanisms for two-sided matching with indifferences. In *Fourth International Workshop on Matching under Preferences*, April 2017. Full version available at https://arxiv.org/abs/1703.10598.

[19] R. Duan and H.-H. Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1413–1424, 2012.

[20] L. E. Dubins and D. A. Freedman. Machiavelli and the Gale-Shapley algorithm. *American Mathematical Monthly*, 88:485–494, 1981.

[21] P. Dütting, M. Henzinger, and I. Weber. An expressive mechanism for auctions on the web. In *Proceedings of the 20th International World Wide Web Conference*, pages 127–136, March 2011.

[22] P. Dütting, M. Henzinger, and I. Weber. Sponsored search, market equilibria, and the Hungarian Method. *Information Processing Letters*, pages 67–73, February 2013.

[23] D. W. Engels, D. R. Karger, S. G. Kolliopoulos, S. Sengupta, R. N. Uma, and J. Wein. Techniques for scheduling with rejection. *Journal of Algorithms*, 49:175–191, 2003.

[24] L. Epstein, J. Noga, and G. J. Woeginger. On-line scheduling of unit time jobs with rejection: minimizing the total completion time. *Operations Research Letters*, 30:415–420, 2002.

[25] A. Erdil and H. Ergin. What's the matter with tie-breaking? Improving efficiency in school choice. *American Economic Review*, 98:669–689, 2008.

[26] A. Erdil and H. Ergin. Two-sided matching with indifferences. Working paper, 2015.

[27] K. Eriksson and J. Karlander. Stable matching in a common generalization of the marriage and assignment models. *Discrete Mathematics*, 217: 135–156, 2000.

[28] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, 1987.

[29] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30: 209–221, 1985.

[30] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[31] F. Glover. Maximum matching in convex bipartite graphs. *Naval Research Logistic Quarterly*, 14:313–316, 1967.

[32] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, pages 287–326, 1979.

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

[34] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities.* Cambridge University Press, 2nd edition, 1952.

[35] N. Kamiyama. A new approach to the Pareto stable matching problem. *Mathematics of Operations Research*, 39:851–862, 2014.

[36] I. Katriel. Matchings in node-weighted convex bipartite graphs. *INFORMS Journal on Computing*, 20:205–211, December 2008.

[37] O. Kesten. School choice with consent. *The Quarterly Journal of Economics*, 125(3):1297–1348, 2010.

[38] D. Knuth. *Marriages Stables.* Montreal University Press, Montreal, 1976.

[39] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[40] H. B. Leonard. Elicitation of honest preferences for the assignment of individuals to positions. *The Journal of Political Economy*, 91:461–479, 1983.

[41] W. Lipski, Jr. and F. P. Preparata. Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems. *Acta Informatica*, 15:329–346, 1981.

[42] D. F. Manlove. *Algorithmics of Matching Under Preferences.* World Scientific, Singapore, 2013.

[43] P. A. Pathak and P. Shi. Simulating alternative school choice options in Boston. Technical report, MIT School Effectiveness and Inequality Initiative, 2013.

[44] C. G. Plaxton. Vertex-weighted matching in two-directional orthogonal ray graphs. In *Proceedings of the 24th International Conference on Algorithms and Computation*, pages 524–534, December 2013.

[45] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[46] M. Quinzii. Core and competitive equilibria with indivisibilities. *International Journal of Game Theory*, 13:41–60, 1984.

[47] L. R. Rabiner and B. Gold. *Theory and application of digital signal processing.* Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975.

[48] A. E. Roth. The economics of matching: Stability and incentives. *Mathematics of Operations Research*, 7:617–628, 1982.

[49] A. E. Roth. The college admissions problem is not equivalent to the marriage problem. *Journal of Economic Theory*, 36:277–288, 1985.

[50] A. E. Roth and E. Peranson. The redesign of the matching market for american physicians: Some engineering aspects of economic design. *The American Economic Review*, 89:748–780, 1999.

[51] A. E. Roth and M. Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis.* Cambridge University Press, New York, 1990.

[52] D. Shabtay, N. Gaspar, and L. Yedidsion. A bicriteria approach to scheduling a single machine with job rejection and positional penalties. *Journal of Combinatorial Optimization*, 23:395–424, 2012.

[53] D. Shabtay, N. Gaspar, and M. Kaspi. A survey on offline scheduling with rejection. *Journal of Scheduling*, 16:3–28, 2013.

[54] L. S. Shapley and H. E. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1:104–123, 1974.

[55] L. S. Shapley and M. Shubik. The assignment game I: The core. *International Journal of Game Theory*, 1:111–130, 1972.

[56] P. Shi. *Prediction and Optimization in School Choice.* PhD thesis, Massachusetts Institute of Technology, 2016.

[57] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32:652–686, July 1985.

[58] S. A. Slotnick. Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research*, 212:1–11, 2011.

[59] S. A. Slotnick and T. E. Morton. Selecting jobs for a heavily loaded shop with lateness penalties. *Computers and Operations Research*, 23:131–140, 1996.

[60] M. Sotomayor. Existence of stable outcomes and the lattice property for a unified matching market. *Mathematical Social Sciences*, 39:119–132, 2000.

[61] M. Sotomayor. The Pareto-stability concept is a natural solution concept for discrete matching markets with indifferences. *International Journal of Game Theory*, 40:631–644, 2011.

[62] G. Steiner and J. S. Yeomans. A linear time algorithm for determining maximum matchings in convex, bipartite graphs. *Computers and Mathematics with Applications*, 31:91–96, 1996.

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