Constructing Stable Matchings Using Preference Elicitation through Prices and Budgets

A thesis presented
by

Lisa Wang

To
The School of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the joint degree of
Bachelor of Arts in Computer Science and Mathematics
Harvard College
Cambridge, Massachusetts

April 1, 2016

Abstract

Eliciting complete preference information is difficult in many two-sided matching markets. First, there may be a very large number of options to rank. In addition, preference elicitation may require interviews that are time and effort costly, and furthermore, costs for elicitation may be unbalanced on the two sides. For example, schools may have easy criteria by which to rank students, but students may need to expend considerable time and energy to rank schools. The problem addressed in this thesis is to achieve a stable matching while minimizing the amount of information required from agents. I focus on the one-to-one two-sided matching problem, or the stable marriage problem, and introduce a new mechanism for achieving a stable matching using virtual prices and virtual budgets. I experimentally analyze the degree of information elicited and the time to convergence, comparing the performance of this new mechanism with two existing stable marriage matching algorithms, Deferred Acceptance and Minimax Regret. The new mechanism requires less information from one side of the market than these other algorithms, which may make it well suited to several real-life market places.

Acknowledgements

I am indebted to Professor David Parkes for introducing me to this topic and serving as my mentor and advisor. I am deeply grateful to him for his tireless and patient guidance throughout this process. I would also like to express my gratitude to Hongyao Ma for sharing her insights and critiquing my research and thesis drafts with such detail and care.

I would also like to thank Professors Yaron Singer and Yiling Chen for generously offering to be my thesis readers.

Finally, thank you to my mother and father for always giving me their unconditional love and support.

Contents

Chapter 1

Introduction

In a two-sided matching market, agents are divided into two disjoint sets. Each agent has a preference for the members on the other side. A matching over all agents is considered stable if no two agents would prefer to leave their assigned match for each other. In a one-to-one matching problem, agents are matched to at most one agent. For example, the stable marriage problem uses ordinal preferences to construct a stable one-to-one matching between a set of men and a set of women. In an extension of one-to-one, the many-to-one matching problem lets agents on one side of the market be matched to multiple agents from the other side. An example is the National Resident Matching Program, in which over 40,000 residents apply to about 5,000 residency programs; a stable matching between graduating medical students and hospitals is computed based on the preferences of both sides [10]. Similarly, in Singapore, Boston, and New York City, stable matchings are computed to assign students to secondary schools [12].

Gale and Shapley [5] introduced the deferred acceptance algorithm, assigning one side of the market as "proposers" and the other as "acceptors," to solve both the stable marriage and the many-to-one problem. They showed that the algorithm computes a stable matching for any complete preference profile and is computationally efficient. However, in most cases, complete, ordinal preference orderings are not necessary to compute a stable matching. Furthermore, as economies scale, in particular for the many-to-one cases, requiring complete information becomes impractical. Consider resident matching in which each resident would have to rank all 5,000 programs and each hospital would have to rank all 40,000 residents. This would be both time and effort intensive, in particular if specific criteria for ranking were not clearly defined. For example, while hospitals might have clear cut-offs in terms of grades or MCAT scores, students may not have such clarity in ranking

their options. Moreover, both the monetary and time cost of interviews would be high for both sides of the market.

Thus, a preference elicitation scheme that minimizes agent interviews is desirable. And while algorithms such as deferred acceptance can be implemented interactively using preference elicitation, they were not designed to minimize elicitation [2]. Much of the work conducted in this area is around refining partial preferences through agent interviews. Rastegari et al. [9] showed that finding a minimal set of partial preferences that supports an optimal matching is NP-hard. Using the assumption that there is correlation in preferences, they use strongly correlated preferences on one side of the market to construct an elicitation scheme that builds on Gale-Shapley, which they call "Lazy Gale-Shapley". Using equivalence classes for preferences rather than a full ordering, the algorithm is shown to converge to an optimal, stable matching for any preference profile. Similarly, Drummond and Boutilier [2] constructed an elicitation scheme that continually refines partial preferences. A stable matching is approximated by minimizing the maximum regret any pair of agents feels in their matching, thus giving the name "minimax regret." For large economies, minimax regret is shown to converge to an approximately stable matching with less preference information than deferred acceptance, as measured by the number of agent interviews. To compare elicitation schemes, different methodologies have been used, including measuring the number of interviews conducted with agents and the types of question asked to agents.

However, since Lazy Gale-Shapley requires strong correlation in preferences and minimax regret approximates stable matchings, it is not clear that the most efficient preference elicitation scheme is among this set of mechanisms. This thesis explores new options, in particular price-based approaches: a price-based, linear programming approach has been successfully used to generate efficient preference elicitation schemes for other matching markets. By using the primal-dual relationship and the dual variables that double as prices, an iterative, ascending price scheme can be used to compute a utility-maximizing allocation without complete preference information. As Parkes [8] shows, this has found success in auctions, in which agents bid on items, and the assignment problem, a one-to-one matching problem in which preferences are cardinal, rather than ordinal. Since the stable marriage problem can also be formulated as a linear program, a similar approach seemed natural. But, while it seems very difficult to find a primal-dual that exposes a natural price structure, inspired by this mechanism, I develop a price-based approach to iteratively elicit information. In particular, by continuously constricting the polytope of possible preference

profiles, an iterative process can be used to converge to a stable matching.

1.1 Contribution

The main contributions of this thesis are as follows:

- I study different primal-dual formulations of the stable marriage problem, and show that the use of ordinal preferences makes understanding the dual variables difficult.
- I develop a price-based approach towards preference elicitation to construct a stable matching. The resulting algorithm is shown to converge to a stable matching for any two-sided one-to-one matching market.
- I present a method of comparing the information collection between algorithms, which measures the size of the polytope of agent preferences upon reaching a stable matching. Under this comparison, the price-based approach requires less information from the proposing side of the market, while needing more information from the accepting side than both deferred acceptance and minimax regret.

The thesis will be structured as follows. Chapter 2 introduces the notation and basic theory of linear programming and the stable matching problem. Chapter 3 describes primal-dual algorithms that exist for other matching markets to show the motivation for exploring this approach. Chapter 4 presents a new price-based approach of preference elicitation, called price-budget alignment, and proves both soundness and convergence. Chapter 5 presents the experimental procedures of testing the different mechanisms. Chapter 6 compares the number of iterations and polytope size between price-budget alignment and both deferred acceptance and minimax regret. Chapter 7 concludes.

Chapter 2

Basic Theory

In constructing a price-based preference elicitation mechanism for finding a stable matching, I first use a linear programming approach. In this chapter, a brief overview of the linear programming theory necessary to understand this approach is given. Basic notation used to understand the stable marriage problem is also provided for easier understanding of the matching mechanisms discussed. In addition, four existing matching mechanisms are detailed, in particular, the contributions from the literature that are relevant to my investigation.

2.1 Linear Programming Theory

A linear program is a maximization or minimization of a linear function subject to a finite number of linear constraints. A standard program takes the following form:

$$\begin{aligned} \max\, & c^T x \\ \text{s.t. } & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where A is an $m \times n$ matrix, x, c are vectors in $\mathbb{R}^n$, and b is a vector in $\mathbb{R}^m$. We define this as the **primal**. The function being maximized is the objective function and the inequalities following are the constraints. A **feasible** solution is one that satisfies all the constraints of the linear program. A linear program is **infeasible** if it has no feasible solutions. To solve the primal is to find a feasible solution for x that maximizes the objective function — this solution is then **optimal**. A linear program is **unbounded** if the optimal solution is unbounded, i.e. $\pm\infty$.

The primal is associated with another linear program, the **dual**:

$$\begin{aligned} &\min b^T y \\ \text{s.t. } &A^T y \geq c \\ &y \geq 0 \end{aligned}$$

where y is a vector in $\mathbb{R}^m$.

Each variable x_j in the primal corresponds to a constraint in the dual. And similarly, each variable y_i in the dual corresponds to a constraint in the primal. We can see that the optimal multiplier y_i corresponds to a change in the primal objective if b_i in the primal increased by 1. The y_i can thus act as "prices" on their corresponding constraints, and are called **shadow prices**.

We use the following two relationships between the primal and its dual.

Theorem 1 (Strong Duality Theorem)**.** *A linear program is feasible and has a finite optimal value if and only if the dual is feasible and has a finite optimal value as well. Furthermore, the optimal values are the same and optimal solutions to both the primal and dual exist.*

Theorem 2 (Complementary Slackness)**.** *A primal and dual solution is optimal if and only if the primal and dual are feasible and satisfy the following complementary slackness conditions:*

- *If a primal variable $x_j > 0$ then the corresponding dual inequality is binding. Similarly, if a dual variable $y_i > 0$ then the corresponding primal inequality is binding.*
- *If a dual inequality is not binding, then the corresponding primal variable is equal to 0. And similarly, if a primal equality is not binding, then the corresponding dual variable is equal to 0.*

2.1.1 Primal-Dual Algorithms

The relationship between the primal and dual can be utilized to solve matching and assignment problems through iterative algorithms. The general structure is to use best-response strategies from agents to calculate the primal and dual solutions. Then, by checking the

complementary slackness conditions, optimality can be verified. By using this method, information can be elicited iteratively as the mechanism runs, rather than requiring complete information at the start.

For example, Parkes [8, 7] uses a primal-dual algorithm to solve the linear program formulation of a combinatorial auction. By interpreting dual variables as prices and payoffs, prices on item bundles can be incrementally increased based on elicited agent information in the form of bids. Optimality can be tested by simply testing the complementary slackness conditions, and the linear program never has to be directly solved.

The primal-dual algorithm takes the following form:

1. Maintain a feasible dual solution of prices.
2. Receive bids from agents for the items based on the prices.
3. Compute a feasible primal solution of item allocation, minimizing violations of complementary slackness conditions given agent input.
4. Terminate if all complementary slackness conditions are satisfied. First, whether an agent is allocated a bundle of items if and only if the bundle has non-negative utility and maximizes utility given the prices. Second, whether the allocation is revenue maximizing given the prices.
5. Adjust the dual solution towards an optimal solution, based on CS conditions and the current primal solution - essentially increasing prices for over-demanded items.

Note that for this method to work, a full understanding of the dual variables, including a price interpretation, is necessary. In particular, steps 1 and 2 require prices to construct the dual solution and to elicit agent information.

2.2 Stable Marriage: Basic Terminology

Throughout this thesis, the following notations are used when describing the stable marriage problem. We start with two disjoint sets, each of size n, denoted P and A, for proposers and acceptors respectively. Each agent p has strictly ordered preferences containing all members of the other set, denoted as $\succ_p$. If agent $p \in P$ prefers agent $a_1 \in A$ to agent $a_2 \in A$ then we use the following notation

$$a_1 \succ_p a_2$$

A matching μ is a bijective correspondence between P and A. If p and a are matched, $p = \mu(a)$ and $a = \mu(p)$. Throughout this thesis, I assume that all agents prefer to be matched to someone rather than remain unmatched. Thus, in a matching μ, all agents are matched to exactly one agent on the other side of the market.

A pair $(p, a) \in P \times A$ is a **blocking pair** in a matching μ if p prefers a to her current partner and a prefers p to her current partner,

$$p \succ_a \mu(a) \quad \text{and} \quad a \succ_p \mu(p)$$

A matching in which there is at least one blocking pair is defined as **unstable** and a matching in which there are no blocking pairs is defined as **stable**.

2.3 Matching Mechanisms

First, we consider the widely referenced Gale-Shapley deferred acceptance algorithm, which has been used as the basis for both the National Resident Matching Program as well as New York public high school matching [10, 1]. We then examine the linear programming approach taken by Vohra [15] as well as the elicitation mechanisms developed by Rastegari et. al [9] and Drummond and Boutilier [2, 3].

To effectively understand and compare these mechanisms, I will use the following running example:

$$\begin{aligned}
&n = 3 \\
&P = \{p_1, p_2, p_3\} \\
&A = \{a_1, a_2, a_3\} \\
&\succ = \{p_1 : [a_2, a_1, a_3] \\
&\qquad p_2 : [a_2, a_3, a_1] \\
&\qquad p_3 : [a_2, a_3, a_1] \\
&\qquad a_1 : [p_2, p_1, p_3] \\
&\qquad a_2 : [p_1, p_3, p_2] \\
&\qquad a_3 : [p_3, p_1, p_2]\}
\end{aligned}$$

where an ordering $[a_i, a_j, a_k]$ means that a_i is the most preferred and a_k is the least preferred.

2.3.1 Gale-Shapley Deferred Acceptance

Gale and Shapley developed an algorithm to find a stable matching by having agents on one side of the market propose to agents of the other. The following describes the algorithm:

1. Let each unmatched proposer p propose to her most preferred choice out of the acceptors who have not rejected p.
2. Each acceptor is tentatively matched to, or "holds," her most preferred proposer to date.
3. Stop when no new proposals are made and match each acceptor to the proposer she is holding.

Gale and Shapley showed that this deferred acceptance algorithm always converges to a stable matching under strict, ordinal preferences of agents in P and A.

The matching can also be considered in the context of optimality. An optimal stable matching μ for agents in P is a matching where every $p \in P$ is matched to her best possible partner out of all possible stable matchings. Conway and Knuth [6] showed that when preferences are strict, if set P proposes in the deferred acceptance algorithm, then the matching is the optimal matching for agents in P and the least optimal for agents in A. Likewise, if the set A proposes, then the matching is the optimal matching for agents in A and the worst for agents in P.

If we consider our running example, let $P(p_i)$ be the agent p_i proposes to and let $A(a_i)$ be the proposer a_i "holds." Then, the rounds proceed as follows:

1. $P(p_1) = a_2, P(p_2) = a_2, P(p_3) = a_2$
   $A(a_2) = p_1$
2. $P(p_2) = a_3, P(p_3) = a_3$
   $A(a_2) = p_1, A(a_3) = p_3$
3. $P(p_2) = a_1$
   $A(a_1) = p_2, A(a_2) = p_1, A(a_3) = p_3$

So, $\mu = \{(a_1, p_2), (a_2, p_1), (a_3, p_3)\}$ and the mechanism took 3 iterations. Note that when a proposer chooses an agent to propose to or when an acceptor chooses her most preferred proposer, preference information is elicited. For example, from the proposals in the first

round, we know that each proposer prefers a_2 to the other two acceptors. And on the acceptor side, we know that a_2 prefers p_1 to both p_2 and p_3. If we let a sublist be an equivalence class, i.e. the order within is not determined, then the information collected is as follows:

$$\begin{aligned}\succ = \{&p_1 : [a_2, [a_1, a_3]] \\ &p_2 : [a_2, a_3, a_1] \\ &p_3 : [a_2, a_3, a_1] \\ &a_1 : [] \\ &a_2 : [p_1, [p_3, p_2]] \\ &a_3 : [p_3, p_2]\}\end{aligned}$$

Note that this is a proposer-optimal matching, and more information was required from the proposers than the acceptors to compute a stable matching in this example..

Elicitation Mechanism

Gale-Shapley can also be implemented using an elicitation approach, such that agents are not required to provide their full preference orders at the start. Incremental queries are used to ask proposers their most preferred choice at each round and acceptors their most preferred proposer rather than drawing from provided preference orders. However it can be seen that deferred acceptance was not constructed to optimize interview minimization. Low-ranked proposers and acceptors will specify close to their entire preference ranking, when only the bottom of their ranking is needed. Still, deferred acceptance provides a general basis to build a preference elicitation mechanism — assigning one side as proposers and the other as acceptors sets up a way to collect information using agent actions. This idea is explored more in Chapter 5.

2.3.2 Dual Ascent Algorithm

Vohra describes a dual ascent algorithm using the linear programming formulation of the stable matching problem. In this algorithm, first, if acceptor a is proposer p's kth choice, then $r_{pa} = n - k$, where n is the number of agents on each side. Then, the goal is to maximize,

$$\max\{\sum_{p \in P} \sum_{a \in A} r_{pa} x_{pa}\} \tag{2.1}$$

where x is a possible stable matching and x_{pa} is an indicator variable of whether p and a are matched. The full linear formulation is detailed in Section 3.2. Let v be the dual variable associated with the blocking constraints for each pair (detailed further in section 3.2.1) and w the dual variable associated with the constraint that ensures each proposer is matched with exactly one acceptor (also detailed in section 3.2.1). Then, we have the following Lagrangian relaxation

$$\begin{aligned} \max & \sum_{p\in P}\sum_{a\in A}[r_{pa} - w_a - v_{pa} - \sum_{k\in A:k\succ_p a} v_{pk} - \sum_{k\in P:k\succ_a p} v_{ka}]x_{kp} \\ s.t. & \sum_{a\in A} x_{pa} = 1, \quad \forall p \in P \\ & x_{pa} \geq 0, \quad \forall (p,a) \in P \times A \end{aligned} \tag{2.2}$$

If we solve this relaxation, then each proposer $p \in P$ should propose to the acceptor $a \in A$ that maximizes

$$r_{pa} - w_a - v_{pa} - \sum_{k\in A:k\succ_p a} v_{pk} - \sum_{k\in P:k\succ_a p} v_{kp}.$$

Let (p^t, v^t) be the values of the multipliers at the start of iteration t and let $(p^0, v^0) = 0$. And let x^t be the optimal choice of x given (p^t, v^t) where $x^t_{pw} = 1$ if p proposers to a. The variables in the dual ascent algorithm Vohra proposes change in the following way:

1. $\hat{p} = 0$ throughout
2. At iteration t, let $\hat{v}_{pa} = x^t_{pa}$ for all $p, a \in P \times A$
3. Set $(p^{t+1}, v^{t+1}) = (p^t, v^t) + (\hat{p}, \hat{v}) = (p^t, v^t) + (0, x^t_{pa})$
4. Set $r^{t+1}_{pa} = r^t_{pa} - v^t_{pa} - \sum_{k\in A:k\succ_p a} v^t_{pk} - \sum_{k\in P:k\succ_a p} v^t_{kp}$.
   In other words let $r^{t+1}_{pa} = r^t_{pa} - x^t_{pa} - \sum_{k\in A:k\succ_p a} x^t_{pk} - \sum_{k\in P:k\succ_a p} x^t_{ka}$.
5. Terminate if x^t is a matching.

When proposers propose, this corresponds to the maximization of the relaxation, given the current Lagrangian multipliers. The responses of the acceptors leads to the updating of the Lagrangian multipliers. And, essentially, if proposer p proposes to acceptor a at iteration t, a is the highest ranked choice of p at that time. And if proposer p is acceptor a's top choice among those proposing to a, then p's value r for a, as well as all acceptors p prefers less than a will decrease by 1. The value of r for all acceptors that p prefers more than a will decrease by at least 1. This ensures at the next iteration, a will still be p's most preferred choice. However, if a rejects p, r_{pa} keeps decreasing in each iteration until

at some point the p finds some other agent more favorable.

We can apply this mechanism to our running example, where the iterations proceed as follows:

1. $r^0 = \begin{bmatrix} 2 & 3 & 1 \\ 1 & 3 & 2 \\ 1 & 3 & 2 \end{bmatrix}, x^0 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
2. $r^1 = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, x^1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
3. $r^2 = \begin{bmatrix} 0 & 1 & -1 \\ -1 & -1 & 0 \\ -1 & -1 & 0 \end{bmatrix}, x^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
4. $r^2 = \begin{bmatrix} -1 & 0 & -3 \\ -2 & -2 & -2 \\ -2 & -2 & -1 \end{bmatrix}, x^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
5. $r^2 = \begin{bmatrix} -2 & -1 & -5 \\ -3 & -4 & -4 \\ -3 & -3 & -2 \end{bmatrix}, x^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Thus we converge to the same matching as deferred acceptance, but with five iterations. We also note that since r requires exact preference ordering from the start, there is no clear way to alter this mechanism to use preference elicitation instead. Also note that while there is repetition in proposing, the general pattern follows deferred acceptance. In fact, Vohra shows that the dual adjusted rank r declines in a way such that proposers will propose in about the same order as they would in deferred acceptance. In particular, the development of this mechanism reflects the motivation to use a linear programming approach in an iterative way.

2.3.3 Lazy Gale Shapley

Rastegari et al. [9] consider the problem of minimizing the number of interviews needed to find a stable matching. They show that finding a minimal set of partial preferences that supports a proposer-optimal matching is NP-hard. They also propose a model in which

agents provide preferences ranked in equivalence classes, where agents in higher-ranked equivalence classes are preferred to those in lower-ranked classes. Since each agent has a true, just unknown, strict preference ordering that is consistent with the equivalence classes, these classes can be refined. For example, $[[p_1, p_2], [p_3, p_4]]$ or $[[p_1, p_2, p_4], [p_3]]$ are equivalence classes consistent with a true ordering $\succ = [p_2, p_1, p_4, p_3]$. A query to elicit information is called an "interview" and reveals pairwise ranking information. The goal is to minimize the number of interviews performed to compute a stable matching.

They define a weakly-dominant interview policy to be one that performs a minimal number of interviews for all underlying true preference profiles consistent with the partial information. A weakly-dominant and sound interview mechanism is described, and can be executed in polynomial time. However, to use this mechanism, which they dub "Lazy Gale-Shapley," identical equivalence classes on one side of the market is necessary. This assumption is the result of observations that preferences are often positively correlated. By using this assumption, the algorithm is then able to converge in polynomial time.

"Lazy Gale-Shapley" operates similarly to Gale-Shapley but rather than requiring complete information at the start, it uses partial ordering in the form of equivalence classes — hence "lazy," and adds an interview mechanism to refine the partial ordering. It should be noted however, that that the algorithm can start with just one equivalence class, with all agents inside, thus providing no initial information and relying completely on the interview elicitation. The interweaving of interview and matching stages are summarized below:

- In the interview stage, one unmatched proposer p that satisfies two properties is chosen. First, there must be achievable acceptors that p has not yet interviewed, where achievable means they have not received a better offer yet — let this set be S. Second, if acceptors rank p in the lth equivalence class, then they rank any other proposer p' that satisfies the first property in a class below or equal to l. p then interviews all acceptors in the highest ranking equivalence class that contains members of S.
- In the matching stage, each unmatched proposer p proposes to her most achievable, interviewed acceptor. Each acceptor accepts her most preferred proposal. The matching phase ends when there are no unmatched proposers that have achievable, interviewed acceptors left.

The algorithm terminates when there are no longer any unmatched employers that still have achievable applicants with whom they have not interviewed. Otherwise, it returns to

the interview stage. The complete algorithm can be found in Appendix A.

We use our running example to illustrate the mechanism. In order to have identical equivalence classes for acceptors, we put all proposers in the same class, so we have $[[p_1, p_2, p_3]]$, and similarly for acceptors. This, allows the algorithm to start with no preference information. The rounds proceed as follows:

1. p_1 interviews with all acceptors. p_1 is matched to a_2 and a_2 is thus no longer achievable for both p_2 and p_3.
2. p_2 interviews with a_1 and a_3. p_2 is matched to a_3.
3. p_3 interviews with a_1 and a_3. p_3 proposes to a_3 who accepts and rejects p_2.
4. p_2 interviews with and is matched to a_1.

The matching is the same proposer-optimal matching as deferred acceptance, and took four iterations of the algorithm to converge. The information collected is as follows:

$$\begin{aligned}\succ = \{&p_1 : [a_2, a_1, a_3] \\ &p_2 : [a_2, a_3, a_1] \\ &p_3 : [a_2, a_3, a_1] \\ &a_1 : [] \\ &a_2 : [p_1, [p_2, p_3]] \\ &a_3 : [p_3, p_2]\}\end{aligned}$$

While it seems that Lazy Gale-Shapley requires more information than deferred acceptance in this instance, the idea of using equivalence classes to reduce information collection is used heavily in both the next algorithm discussed and the mechanism introduced in Chapter 4.

2.3.4 Minimax Regret

Drummond and Boutilier (2013) similarly use the idea of equivalence classes as a basis for their regret-based halving elicitation method, in which agents split a block of preferences into a more preferred half and less preferred half. This method of refining preferences also plays a role in the elicitation mechanism introduced in Chapter 4. They also focus on approximating a stable match by minimizing the instability of any given pair under a matching.

Drummond and Boutilier define several concepts to describe instability. Regret for an agent is defined as the possible improvement in rank position of a new partner. Instability of an unmatched pair (p, a) under a matching μ is the minimum of the potential regret each would feel by choosing their matched partner under μ over each other, taken over all possible preference profiles. Instability of a match is the maximum instability over all possible pairs, and the goal is to minimize this value. So, a matching is chosen that minimizes the maximum regret (ties are broken with randomization) — thus giving the name minimax regret.

The algorithm runs minimax regret to find a matching μ, interviews the agents contributing to the max regret of μ to refine the partial preferences, then repeats, until the minimum maximum regret is below some threshold. For small economies, finding a suitable threshold is impossible. Any viable threshold (i.e. convergence under all preferences profiles) admits both stable and unstable matchings. To better understand, consider our running example once again. For full understanding of how the blocks that are split on each round are chosen, refer to the full algorithm in Appendix A. With no initial information, every matching has the same regret, so we choose one at random and the iterations proceed as follows:

1. $\mu = \{(p_1, a_3), (p_2, a_2), (p_3, a_1)\}$. Minimum maximum regret is 2, and every agent contributes a possible blocking pair that has regret 2. Every agent is thus asked to halve their preferences with the following result:

$$\begin{aligned} \succ = \{&p_1 : [[a_1, a_2], a_3] \\ &p_2 : [[a_2, a_3], a_1] \\ &p_3 : [[a_2, a_3], a_1] \\ &a_1 : [[p_1, p_2, p_3]] \\ &a_2 : [[p_1, p_2, p_3]] \\ &a_3 : [[p_1, p_2, p_3]]\} \end{aligned}$$

2. $\mu = \{(p_1, a_3), (p_2, a_1), (p_3, a_2)\}$. Minimum maximum regret is 2, and agents p_1 and p_2 contribute possible blocking pairs with regret 2. Agents are asked to halve their

preferences with the following result:

$$\begin{aligned}\succ = \{&p_1 : [[a_1, a_2], a_3] \\ &p_2 : [[a_2, a_3], a_1] \\ &p_3 : [[a_2, a_3], a_1] \\ &a_1 : [[p_1, p_2, p_3]] \\ &a_2 : [[p_1, p_2], p_3] \\ &a_3 : [[p_1, p_3], p_2]]\}\end{aligned}$$

3. $\mu = \{(p_1, a_1), (p_2, a_2), (p_3, a_3)\}$. Minimum maximum regret is 2, and agents p_1 and p_2 contribute possible blocking pairs with regret 2. Agents are asked to halve their preferences with the following result:

$$\begin{aligned}\succ = \{&p_1 : [a_2, a_1, a_3] \\ &p_2 : [a_2, a_3, a_1] \\ &p_3 : [[a_2, a_3], a_1] \\ &a_1 : [[p_1, p_2, p_3]] \\ &a_2 : [[p_1, p_2], p_3] \\ &a_3 : [[p_1, p_3], p_2]]\}\end{aligned}$$

4. $\mu = \{(p_1, a_2), (p_2, a_1), (p_3, a_3)\}$ is the resulting stable matching.

To measure success, Drummond and Boutilier use the number of iterations and number of queries to compare with existing algorithms. However, this methodology does not take into account the different amounts of information different queries elicit. For example, eliciting the following pair-wise orders,

$$(1, 2), (1, 3)$$

provides less information than eliciting

$$(1, 2), (2, 3).$$

They attempt to address the problem by using different types of interviews. Drummond and Boutilier [**?**] extend their minimax regret algorithm to take into consideration the idea that the closer together two options are in an agent's preference, the more difficult they are to compare. They create distinctions between a "query" and an "interview." A halving "query" as in above, is said to take less cognitive effort than multiple pair wise comparisons,

or an "interview." The reasoning for this is not entirely convincing, and thus, we focus more on the original minimax regret algorithm throughout this paper for comparison, and present an alternative method of measuring information collection in Chapter 5.

Chapter 3

Linear Program Formulations of Markets

As mentioned, many market based problems involve maximizing some linear function of decision variables. This can be seen in auctions, where the auctioneer is either maximizing revenue or social welfare and the agents are maximizing utility. Because the corresponding dual of the linear program has intuitive interpretations of prices and pay-offs, a primal-dual algorithm can be constructed. As described in Chapter 2, such a mechanism converges to the optimal solution of a linear program without having to solve it directly. In particular, for market problems, the primal-dual mechanism allows for incremental preference elicitation, removing the need for providing complete information at the outset. This chapter demonstrates the use of this mechanism and explores the possibility of using it to solve the stable marriage problem.

3.1 Single Item Auction

The simplest example is the single item auction. The economy has one item and a set N of n agents. Intuitively, the auctioneer maximizes the welfare of the agents by allocating the item to the agent who values it the most.

3.1.1 LP and Dual

We can write the primal of the LP formulation as follows:

$$\begin{aligned} \max & \sum_{i \in N} x_i v_i \\ \text{s.t.} & \sum_{i \in N} x_i \leq 1 \\ & x_i \geq 0, \quad \forall i \in N \end{aligned}$$

where x_i is an indicator variable representing whether or not bidder i gets the item and v_i is the value bidder i would get from the item. The dual is then

$$\begin{aligned} & \min p \\ \text{s.t.} \ & p \geq v_i, \quad \forall i \in N \\ & p \geq 0 \end{aligned}$$

where the dual variable p has a natural interpretation as a price.

3.1.2 Complementary Slackness

The complementary slackness theorem implies that for an optimal allocation of the item, the following must be true:

1. $p > 0 \implies \sum_{i \in N} x_i = 1$
2. $x_i > 0 \implies p = v_i$

By the first implication, if the price is non-zero then the item has been sold. And by the next implication, if the item is sold to bidder i, the price must be equal to the value bidder i ascribes to the item. This follows the intuitive logic of a single-item auction. Thus, if any proposed feasible allocation does not follow these implications, it is not an optimal allocation.

If we consider a primal-dual algorithm to determine the optimal allocation, we start with the simplest feasible dual solution of $p = \max v_i$. But because there are not explicit pay-off variables for agents, determining subsequent agent actions is not straightforward. However, we can add in redundant constraints to induce payoff variables in the dual, such that the

primal is as follows:

$$\begin{aligned} \max & \sum_{i \in N} x_i v_i \\ \text{s.t.} & \sum_{i \in N} x_i \leq 1 \\ & x_i \leq 1, \quad \forall i \in N \\ & x_i \geq 0, \quad \forall i \in N \end{aligned}$$

The dual then has an additional n variables that can be interpreted as profits:

$$\begin{aligned} \min & \, p + \sum_{i \in N} \pi_i \\ \text{s.t.} & \; p + \pi_i \geq v_i, \quad \forall i \in N \\ & p \geq 0 \end{aligned}$$

Now the complementary slackness conditions are richer:

1. $p > 0 \implies \sum_{i \in N} x_i = 1$
2. $\pi_i > 0 \implies x_i = 1$
3. $x_i > 0 \implies p + \pi_i = v_i$

In particular, the interpretation of the third condition is that if the item is allocated to agent i, then her pay-off is the value she gains from the item, v_i, minus the price she pays, p_i.

We note that the optimal dual conditions are

$$\pi_i = \max(0, v_i - p)$$

which allows for minimization of the dual objective as well as feasibility. A primal-dual algorithm could start with the feasible dual solution of $p = 0, \pi_i = \max(0, v_i - p) = v_i$. A feasible primal would then assign the item to some agent x_j who has $\pi_j > 0$, but then most of the complementary slackness conditions in Category 2 are violated. So, we increase the price to $p = 1$ and decrease profits to $\pi_i = \max(v_i - p, 0)$. We continually repeat this process until only one agent x_j has $\pi_j > 0$. At this point, complementary slackness conditions are all satisfied and we have converged to the optimal allocation.

For more examples of primal-dual process, such as the primal-dual formulation of single-sided assignment and two-sided assignment, see Appendix B.

3.2 Marriage Problem

We now consider the linear formulation of the stable marriage problem. Important to note is that in the Gale-Shapley model, preferences are purely ordinal, rather than cardinal, such as in the auction problem. In addition, there is no utility transfer between agents. A model in which there are transfers, also known as the two-sided assignment problem, can be found in Appendix A. When preferences are cardinal, dual variables can be easily interpreted as payoff vectors for agents, and the optimal dual conditions and complementary slackness conditions can be easily interpreted and checked. However, in the stable marriage problem, the ordinal preferences makes a dual variable interpretation more difficult.

3.2.1 LP and Dual

As noted before, we assume the same number of proposers and acceptors, n, and all agents prefer to be matched than to be single. A matching is represented by a matrix $x \in [0,1]^{n \times n}$ where $x_{p,a} = 1$ if proposer p and acceptor a are matched. Vande Vate [13] showed that a reasonable objective is to maximize the number of matchings. Constraints 3.2 and 3.3 ensure that each agent is matched to at most one other agent on the other side. Constraint 3.4 ensures that there are no blocking pairs, in other words, for any pair $(p,a) \in P \times A$ either agent p is matched to someone she prefers to agent a, or agent a is matched to someone she prefers to agent p, or they are matched with each other. These constraints ensure stability. Gale and Shapley [5] showed that the solution space for this primal is nonempty.

The primal is as follows:

$$\max \sum_{(i,j) \in P \times A} x_{i,j} \tag{3.1}$$

$$\text{s.t.} \sum_{j \in A} x_{p,j} \leq 1, \quad \forall p \in P \tag{3.2}$$

$$\sum_{i \in P} x_{i,a} \leq 1, \quad \forall a \in A \tag{3.3}$$

$$\sum_{j \succ_p a} x_{p,j} + \sum_{i \succ_a p} x_{i,a} + x_{p,a} \geq 1, \quad \forall (p,a) \in P \times A \tag{3.4}$$

$$x_{p,a} \geq 0, \quad \forall (p,a) \in P \times A$$

The dual for this primal is as follows:

$$\begin{aligned} \min & \sum_{i \in P} \alpha_i + \sum_{j \in A} \beta_j - \sum_{i \in P} \sum_{j \in A} \gamma_{i,j} \\ \text{s.t. } & \alpha_p + \beta_a - \sum_{j \prec_p a} \gamma_{p,j} - \sum_{i \preceq_a p} \gamma_{i,a} \geq 1, \quad \forall (p,a) \in P \times A \\ & \alpha, \beta, \gamma \geq 0 \end{aligned} \tag{3.5}$$

From Roth et al. [11] we know that given a solution x to the primal, the following is a corresponding dual solution:

$$\begin{aligned} \alpha_p &= \sum_{j \in A} x_{p,j}, \quad \forall p \in P \\ \beta_a &= \sum_{i \in P} x_{i,a}, \quad \forall a \in A \\ \gamma_{p,a} &= x_{p,a}, \quad \forall (p,a) \in P \times A \end{aligned}$$

And if the solution x is optimal, then the above dual solution is also optimal. However, a primal-dual mechanism is still not clear, so we turn to the complementary slackness conditions for better understanding.

3.2.2 Complementary Slackness

The conditions for optimal solutions are as follows:

1. $x_{p,a} > 0 \implies \alpha_p + \beta_a - \sum_{j \succ_p a} \gamma_{p,j} - \sum_{i \succeq_a p} \gamma_{i,a} = 1$
2. $\alpha_p > 0 \implies \sum_{j \in A} x_{p,j} = 1$
3. $\beta_a > 0 \implies \sum_{i \in P} x_{i,a} = 1$
4. $\gamma_{p,a} > 0 \implies \sum_{j \succ_p a} x_{p,j} + \sum_{i \succ_a p} x_{i,a} + x_{p,a} = 1$

First we note that none of the dual variables have an immediate, clear interpretation as prices or pay-offs. In addition, the high degree of freedom that the dual variables have makes any interpretation difficult. To better understand, consider our running example

from Chapter 2. To refresh,

$$\begin{aligned} n &= 3 \\ P &= \{p_1, p_2, p_3\} \\ A &= \{a_1, a_2, a_3\} \\ \succ &= \{p_1 : [a_2, a_1, a_3] \\ &\quad p_2 : [a_2, a_3, a_1] \\ &\quad p_3 : [a_2, a_3, a_1] \\ &\quad a_1 : [p_2, p_1, p_3] \\ &\quad a_2 : [p_1, p_3, p_2] \\ &\quad a_3 : [p_3, p_1, p_2]\} \end{aligned}$$

Our optimal primal solution is

$$x = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The complementary slackness constraints on the corresponding optimal dual are:

1. $\alpha_1 + \beta_2 - \gamma_{1,2} = 1$
2. $\alpha_2 + \beta_1 - \gamma_{2,2} - \gamma_{2,3} - \gamma_{2,1} = 1$
3. $\alpha_3 + \beta_3 - \gamma_{3,2} - \gamma_{3,3} = 1$

We first note that we can set all $\alpha_i = \beta_j = 1$ and $\gamma_{p,a} = x_{p,a}$ as previously discussed, to get an optimal dual solution and optimal value of 3. However, we also have the optimal solution all $\alpha_i = 1$ and all $\beta_j = \gamma_{i,j} = 0$, which satisfies the constraints of the dual and the complementary slackness conditions. This level of potential variability makes an intuitive interpretation of the dual variables unclear and in particular, none of the dual variables resemble prices or payoffs.

3.3 Introducing Redundancy in Primal Constraints

As in the auction, we can add redundancy to the linear program in order to gain additional insight into the problem. Similarly, we add the constrain that all $x_{p,a} \leq 1$.

3.3.1 LP and Dual

The primal is as follows:

$$\begin{aligned} \max \sum_{(i,j)\in P\times A} x_{i,j} \\ \text{s.t.} \sum_{j\in A} x_{p,j} \leq 1, \quad \forall p \in P \\ \sum_{i\in P} x_{i,a} \leq 1, \quad \forall a \in A \\ \sum_{j \succ_p a} x_{p,j} + \sum_{i \succ_a p} x_{i,a} + x_{p,a} \geq 1, \quad \forall (p,a) \in P \times A \\ x_{p,a} \leq 1, \quad \forall (p,a) \in P \times A \\ x_{p,a} \geq 0, \quad \forall (p,a) \in P \times A \end{aligned} \tag{3.6}$$

The dual is as follows:

$$\begin{aligned} \min \sum_{i\in P} \alpha_i + \sum_{j\in A} b_j - \sum_{i\in P}\sum_{j\in A} \gamma_{i,j} + c_{i,j} \\ \text{s.t. } \alpha_p + \beta_a - \sum_{j \prec_p a} \gamma_{p,j} - \sum_{i \preceq_a p} \gamma_{i,a} + c_{p,a} \geq 1, \quad \forall (p,a) \in P \times A \\ \alpha, \beta, \gamma, p \geq 0 \end{aligned} \tag{3.7}$$

3.3.2 Complementary Slackness

From this primal-dual model, we have the following complementary slackness conditions:

1. $x_{p,a} > 0 \implies \alpha_p + \beta_a - \sum_{j \succ_p a} \gamma_{p,j} - \sum_{i \succeq_a p} \gamma_{i,a} + c_{p,a} = 1$
2. $\alpha_p > 0 \implies \sum_{j\in A} x_{p,j} = 1$
3. $\beta_a > 0 \implies \sum_{i\in P} x_{i,a} = 1$
4. $\gamma_{p,a} > 0 \implies \sum_{j \succ_p a} x_{p,j} + \sum_{i \succ_a p} x_{i,a} + x_{p,a} = 1$
5. $c_{p,a} > 0 \implies x_{p,a} = 1$

The $c_{p,a}$ variables do indeed have a price interpretation — if $c_{p,a}$ is positive, then agents p, a must be matched, and if they are not matched, then $c_{p,a}$ must be 0. However, even so, the other dual variables still have high degrees of freedom and are not easily interpretable, particularly as prices or pay-offs.

3.4 Alternative Linear Formulation

We note that even if there were a clear interpretation of prices and pay-offs in the linear formulation, the "affordability" of an acceptor a is not only based on the demand for a but also on the preferences of a. A suitable price mechanism then, should include budgets for the proposers that reflect the preferences of the acceptors. Because preferences are ordinal, the actual values of prices and budgets do not matter as long as they reflect the preference orderings. For example, if acceptor a_1 is the first choice of all the proposers, the price of a_1 should be higher than the prices for other acceptors, but the actual price value is irrelevant. Ideally, under some set of prices and budgets, if each proposer is matched to the acceptor she most prefers out of the ones she can afford, then this results in a stable matching.

To construct such prices and budgets, we use r_i for each $i \in A$ and budgets $b_{i,j}$, for each $i, j \in P \times A$ that represent the budget proposer i has for acceptor j. Intuitively, prices are based on how "demanded" acceptors are, and budgets are based on the preferences of the acceptors. Agent-specific budgets allow for more flexible budgets in response to both prices and acceptor preferences. This price and budget structure is the basis for both the linear formulation in this section as well as for the mechanism constructed in Chapter 4.

3.4.1 Linear Program

So, given a matching μ and preference orders $\succ$, if proposer p is matched to acceptor a, then p should be able to afford a. And if proposer p is matched to acceptor a but prefers acceptor a', p should not be able to afford a'. Furthermore, variation in proposer p's different budgets should be minimized in order to better reflect a true market. These constraints result in the following linear program, which we will refer to as the Price LP.

$$\begin{aligned} \min\ & z - y && \text{(Price LP)} \\ \text{s.t. } & c_j \leq b_{i,j}, && \text{if } \mu(i) = j \\ & c_j > b_{i,j}, && \text{if } j \succ_i \mu(i) \\ & b_{i,j} - b_{i,k} \geq y, && \forall j, k \in A, i \in P \\ & b_{i,j} - b_{i,k} \leq z, && \forall j, k \in A, i \in P \\ & c, b, z, y \geq 0 \end{aligned}$$

Note, that when implementing this linear program, the second constraint, which uses a strict inequality, must be changed to

$$c_j \geq \epsilon + b_{i,j}, \quad \text{if } j \succ_i \mu(i)$$

Consider the running example from Chapter 2 once more, using this LP and $\epsilon = 0.05$, the price and budget matrices are

$$c = \begin{bmatrix} 0 & 0.05 & 0.05 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 & 0.05 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}$$

We can see that the only proposer that can afford a_2 is b_1 and the only proposer that can afford a_3 is b_3, and the only acceptor b_2 can afford is a_1. This results in the stable matching achieved through the previous mechanisms and follows from the following theorem:

Theorem 3 (Proof of Price and Budget Soundness)**.** *Given a stable matching μ and preference order $\succ$, if the prices and budgets resulting from the Price LP are used such that proposers are matched with their most preferred, affordable acceptor, then μ is the resulting match.*

Proof: Let μ be stable matching for preference profile $\succ$. Suppose the prices resulting from the Price LP are $\boldsymbol{c}, \boldsymbol{b}$ and that if proposers are matched with their most preferred, affordable acceptor, a matching $\mu' \neq \mu$ results. This implies that there exists a matched pair $(p, a) \in \mu'$ that does not exist in μ. First, if (p, a) both prefer the matching in μ' over μ, (p, a) would be a blocking pair of μ, which cannot be true since μ is stable. So either only p prefers μ' or only a prefers μ'. If p prefers a to $\mu(p)$, then $c_a > b_{p,a}$, so they could not be matched in μ'. If a prefers p to $\mu(a)$, p cannot prefer a to $\mu(p)$, and since p can still afford $\mu(p)$, (p, a) would not be matched in μ'. Thus, we have a contradiction, so $\mu = \mu'$.

3.4.2 Proposed Mechanism

A proposed iterative algorithm initializes prices and budgets to 0 and guesses a preference profile for the participating agents based on the prices and budgets. Using this preference profile, a matching is simulated and fed into the Price LP to get new prices and budgets. The proposers are queried as to whether, under the new prices and budgets and true preferences, they would prefer to be matched to a different agent. If all proposers are happy, then terminate. If proposers prefer otherwise, then update the preferences and repeat.

Algorithm 1

procedure ALGORITHM 1(P, A)
  $c[j] \leftarrow 0$ for $j \in A$
  $b[i, j] \leftarrow 0$ for $(i, j) \in P \times A$
  randomly initialize $\succ$
loop:
  input $\succ$ into deferred acceptance to compute stable matching μ
  input μ into the Price LP to get new c, b
  for $p \in P$ **do**
    if under c, b and true preferences, i prefers an acceptor $a \neq \mu(p)$ **then**
      update $\succ$ to reflect $a \succ_p \mu(p)$
  terminate if $\succ$ unchanged and return μ

We can again use the running example to understand how this algorithm works. Recall,

$$\begin{aligned} n &= 3 \\ P &= \{p_1, p_2, p_3\} \\ A &= \{a_1, a_2, a_3\} \\ \succ &= \{p_1 : [a_2, a_1, a_3] \\ &\quad\ p_2 : [a_2, a_3, a_1] \\ &\quad\ p_3 : [a_2, a_3, a_1] \\ &\quad\ a_1 : [p_2, p_1, p_3] \\ &\quad\ a_2 : [p_1, p_3, p_2] \\ &\quad\ a_3 : [p_3, p_1, p_2]\} \end{aligned}$$

has stable matching $\{(p_1, a_2), (p_2, a_1), (p_3, a_3)\}$. The algorithm proceeds as follows:

1. Guess $\succ$ to get $\mu = \{(p_1, a_1), (p_2, a_2), (p_3, a_3)\}$. Using the Price LP, we find

$$c = \begin{bmatrix} 0 & 0.05 & 0.05 \end{bmatrix}, \quad b = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}$$

   Under these prices and budgets, the proposers are with their most preferred, affordable acceptor, so the algorithm terminates.

However, μ is not a stable matching, as (p_1, a_2) is a blocking pair.

Prices and budgets were theorized to converge towards reflecting the true preferences. Under this assumption, proposers would only be happy if they were matched to their most preferred, affordable acceptor, which would imply the matching is stable. Because the prices and budgets reflect guessed preferences, proposers might still be able to afford agents they truly preferred to their match. However, in some circumstances, as above, the proposers are not able to afford any acceptors they actually prefer to their match. In these cases, the algorithm terminates with an unstable matching.

Chapter 4

Iterative Price and Budget Algorithm for Preference Elicitation

While the linear formulation was not a successful approach, it raised the idea of using prices and budgets to elicit information from agents, in particular asking proposers to identify their most preferred, affordable acceptor. Drawing on the method of refining preferences in the form of equivalence classes, as raised by Drummond and Boutilier [2] and Rastegari et al. [9] as described in Chapter 2, prices and budgets can be refined similarly. In this chapter, these two methods are combined to construct an iterative, elicitation mechanism for computing a stable matching.

4.1 Algorithm

In this mechanism, the same structure of prices and budgets are used as in Section 3.4, where each acceptor is given a price c_a and each proposer has acceptor-specific budgets $b_{p,a}$. Each proposer is matched to her most preferred, affordable acceptor. If each acceptor is matched to one proposer, the algorithm terminates, returning the matching. Otherwise, the price of any over-demanded acceptor is increased. If no proposers can afford the acceptor at this new price, the acceptor is asked to identify the proposal she most prefers. The acceptor then splits her preference into a block she prefers at least as much as this proposer, and one she prefers less. The budget of each proposer in the more preferred block is increased, and the mechanism repeats. Since the algorithm is essentially incrementally aligning prices

and budgets with agent preferences to achieve a stable matching, the mechanism will be referred to as price-budget alignment.

Algorithm 2

procedure PRICE-BUDGET ALIGNMENT(P, A, k)
  $c_a \leftarrow 0, \quad \forall a \in A$
  $b_{p,a} \leftarrow 0, \quad \forall (p, a) \in P \times A$
loop
  for $p \in P$ **do**
    $B_p \leftarrow \{a | a \in A, c_a \leq b_{p,a}\}$
    $m(p) \leftarrow$ most preferred $a \in B_p$
  if m is bipartite **then**
    return m
  for $a \in A$ **do**
    $D_a = \{p | m(p) = a\}$
    if $|D_a| > 1$ **then**
      $c_a \leftarrow c_a + \delta$
      if $\nexists p \in P$ s.t. $c_a \leq b_{p,a}$ **then**
        $p' \leftarrow a$'s most preferred in D_a
        for $p \in P$ **do**
          if $p \succeq_a p'$ **then**
            $b_{p,a} \leftarrow b_{p,a} + \delta$

As we can show below, this procedure converges to a stable, proposer-optimal matching for any sets P, A.

4.2 Convergence, Soundness and Optimality

Theorem 4 (Convergence)**.** *Given two disjoint sets of agents, each of size n, where each agent has strict, ordinal preferences over the other set, price-budget alignment converges to a matching μ.*

Proof: First note that at the beginning of each iteration k, at least one acceptor is over-demanded, otherwise the algorithm must have terminated. For each acceptor a who is over-demanded, the price of a will increase. If p is the most preferred proposer of a among those who proposed to a during iteration k, then an identical increase in budget will occur

for agents a prefers at least as much as p. So, at least one proposer will still be able to afford a. However, less preferred proposers will no longer be able to afford a, and the number of proposers who can afford a will decrease after iteration k. Thus, since the number of proposers who can afford each acceptor can never go below one, the algorithm must terminate and converge to a matching. □

Theorem 5 (Soundness)**.** *The matching that price-budget alignment converges to is stable.*

Proof: Let μ be the matching the algorithm terminates at. Assume that there is a blocking pair (p, a). If p can afford a, then they would be matched, so it must be that p cannot afford a. But if p cannot afford a, then a must have been proposed to, and subsequently matched to, a more preferred proposer p', causing the price of a to increase without an equivalent increase in the budget of p for a. So (p, a) is not a blocking pair, and we have a contradiction. □

We can also show that price-budget alignment converges to the proposer-optimal matching, using the idea behind the Conway, Knuth [6] proof.

Theorem 6 (Proposer-Optimal)**.** *The matching that price-budget alignment converges to is proposer-optimal.*

Proof: We say that a is achievable for p if p, a are matched in some stable matching. We can show by induction that no acceptor a is even unaffordable to a proposer p if a is achievable for p. This is trivially true at iteration 1, because prices and budgets are all 0. Suppose that this is true through iteration k of the algorithm. Then, suppose at iteration $k + 1$, an achievable acceptor a is unaffordable to proposer p. This would imply there is some preferred proposer p' who proposed to a during iteration k, thereby increasing the price of a, but not the budget of p for a. Since a is achievable for p, p, a are matched in some stable matching μ. Then under μ, p' must be matched to some a' she prefers, otherwise (p', a) is a blocking pair of μ. However, since a' is achievable for p, then at round k of the algorithm, a' is affordable for p, so p will have proposed to a' rather than a, which contradicts our assumption. □

4.3 Running Example

We can use our running example to better illustrate this mechanism:

$$
\begin{aligned}
&n = 3 \\
&P = \{p_1, p_2, p_3\} \\
&A = \{a_1, a_2, a_3\} \\
&\succ = \{p_1 : [a_2, a_1, a_3] \\
&\qquad p_2 : [a_2, a_3, a_1] \\
&\qquad p_3 : [a_2, a_3, a_1] \\
&\qquad a_1 : [p_2, p_1, p_3] \\
&\qquad a_2 : [p_1, p_3, p_2] \\
&\qquad a_3 : [p_3, p_1, p_2]\}
\end{aligned}
$$

The mechanism proceeds in the following iterations:

1. All proposers match to a_2. So, c_2 updates to 0.05 and $b_{2,2}$ updates to 0.05.
2. p_2 matches to a_2 and p_1, p_3 match to a_3. So, c_3 updates to 0.05 and $b_{3,3}$ updates to 0.05
3. p_2 matches to a_3, p_1 matches to a_2 and p_3 matches to a_3, so the algorithm terminates.

This is the same stable matching we previously found, and the algorithm converged after three iterations, with the following information collected:

$$
\begin{aligned}
&\succ = \{p_1 : [a_2, [a_1, a_3]] \\
&\qquad p_2 : [a_2, a_3, a_1] \\
&\qquad p_3 : [a_2, a_3, a_1] \\
&\qquad a_1 : [] \\
&\qquad a_2 : [p_1, [p_3, p_2]] \\
&\qquad a_3 : [p_3, p_2]\}
\end{aligned}
$$

We compare this to the information collected by deferred acceptance:

$$\begin{aligned}\succ = \{&p_1 : [a_2, [a_1, a_3]] \\ &p_2 : [a_2, a_3, a_1] \\ &p_3 : [a_2, a_3, a_1] \\ &a_1 : [] \\ &a_2 : [p_1, [p_3, p_2]] \\ &a_3 : [p_3, p_2]\}\end{aligned}$$

and the information collected by Lazy Gale-Shapley:

$$\begin{aligned}\succ = \{&p_1 : [a_2, a_1, a_3] \\ &p_2 : [a_2, a_3, a_1] \\ &p_3 : [a_2, a_3, a_1] \\ &a_1 : [] \\ &a_2 : [p_1, [p_2, p_3]] \\ &a_3 : [p_3, p_2]\}\end{aligned}$$

and the information collected by minimax regret:

$$\begin{aligned}\succ = \{&p_1 : [a_2, a_1, a_3] \\ &p_2 : [a_2, a_3, a_1] \\ &p_3 : [[a_2, a_3], a_1] \\ &a_1 : [] \\ &a_2 : [[p_1, p_2], p_3] \\ &a_3 : [[p_1, p_3], p_2]]\}\end{aligned}$$

We see immediately that less information is collected than Lazy Gale-Shapley, and the same amount as deferred acceptance. The comparison between minimax regret is less clear. This sort of uncertainty motivates the comparison analysis in Chapter 5.

Chapter 5

Experimental Methodology

In this chapter, the experimental environment is described, including the methods used to generate preferences and compare algorithm performance. As seen in the comparison in Section 4.3, comparisons of preference elicitation can be tricky due to the different possible measurements available, so a method of computing the size of the agent polytope of preferences is described.

5.1 Preference Generation

In testing the different mechanisms, we use uncorrelated, weakly correlated, strongly correlated, and for some comparisons, perfectly correlated preferences. Using different levels of correlation allows comparison of the different mechanisms under different possible economies. It is possible to imagine a market where interests might be highly correlated, such as high-school matching for a more homogenous population, versus a market where interests are weakly correlated, such as residency matching, versus a market where interests are uncorrelated, such as matching internet advertisements to users [4].

Generating uncorrelated preferences is straightforward, simply randomize for each agent. And for generating perfectly correlated preferences, we use the same randomized order for all agents. To generate weakly correlated preferences for n agents, we used the following method:

1. Choose an ordering $\succ_1$ uniformly at random for agent 1. Let $r_{\succ_1}(i)$ be the ranking of agent i in agent 1's preference order.

2. Set the distribution of the preferences to be such that

$$P(i) = \frac{n + 1 - r_{\succ_1}(i)}{\sum_{k=1}^{n} k}$$

3. For agents 2...n, choose an ordering without replacement using the distribution P.

For example, suppose $\succ_1 = (2, 1, 3, 4)$, then $p = [0.3, 0.4, 0.2, 0.1]$. When we generate preferences for each agent, we use p to "draw" agents without replacement to formulate an ordering. So when we generate $\succ_2$, we choose the most preferred agent using p. Suppose we choose agent 1 first, then we remove the probability for agent 1 from p and normalize the distribution, so $p = [0, 4/7, 2/7, 1/7]$. Then we choose again using p, repeating until a full preference order $\succ_2$ is generated.

To generate strongly correlated preferences, we use the same method, except in step 2, we set the distribution to be

$$P(i) = 0.8(0.2)^{i-1}, \quad \text{if } i \neq n \tag{5.1}$$

$$P(i) = 1 - \sum_{k=1}^{n-1} P(k), \quad \text{if } i = n \tag{5.2}$$

5.2 Collection Process

We store preference information as ordered pairs, so if the following information,

$$[(a_1, a_2), (a_1, a_3)]$$

has been collected for agent p_1, we know that the agent p_1 prefers a_1 to both a_2 and a_3. To get the possible full rankings that abide by the information collected, we use the algorithm described in Varol and Rotem [14] that uses topological sort to find all the complete orderings that are possible given a partial order.

5.3 Measuring Preference Elicitation

Drummond and Boutilier [2] compute the number of iterations to convergence and the number of interviews required to approximate a stable matching in order to compare between minimax regret and deferred acceptance. While the number of iterations required gives a fair estimate on the time required for the mechanism to converge, using the number

of interviews to measure preference elicitation has a few problems. As discussed in Section 2.3.4, different queries provide varying amounts of information.

In this analysis, the polytope of possible preference profiles after the elicitation mechanism completes is used for comparison — the larger the polytope, the less information collected from agents. The size of the polytope is calculated by enumerating the preferences profiles that fit the information collected, and normalizing over the set of all possible preference profiles. As the number of ordered pairs of information collected increases, the number of preference profiles that match the information collected decreases, and thus the size of the polytope decreases. Thus, the size of the polytope is an effective measure of the information collected. Below, the methodology for calculating the polytope size is described.

5.3.1 Profile Enumeration

To calculate the size, we simply take the fraction of the number of preference profiles across agents in the polytope over the total number of possible profiles. So if we have three proposers where the possible rankings for each are

$$\begin{aligned}
\succ_{p_1} &\in \{[a_1, a_3, a_2], [a_1, a_2, a_3]\} \\
\succ_{p_2} &\in \{[a_2, a_1, a_3]\} \\
\succ_{p_3} &\in \{[a_1, a_2, a_3], [a_1, a_3, a_2]\}
\end{aligned}$$

And we have three acceptors where the possible rankings for each are

$$\begin{aligned}
\succ_{a_1} &\in \{[p_2, p_3, p_1]\} \\
\succ_{a_2} &\in \{[p_3, p_2, p_1], [p_3, p_1, p_2]\} \\
\succ_{a_3} &\in \{[p_1, p_2, p_3]\}
\end{aligned}$$

Then there are a total of 8 different complete preference profiles for this economy. The total number of possible profiles is

$$(n!)^{2n} = (3!)^6 = 46656$$

So the size of the polytope is

$$\frac{8}{46656} = 1.71 \times 10^{-4}$$

5.3.2 Alternative Method

For cases where preferences are correlated, we can consider the probability of each preference profile when we calculate the polytope. Let f_p be the PDF for preference profiles for proposers and let f_a be the PDF for acceptors. Let X_p, X_a be the set of all possible preference profiles for n proposers and n acceptors respectively. Finally, for $i \in X_p$, let $I_p(i) = 1$ if x is in the polytope and 0 otherwise, and similarly for acceptors. Then, we note that the probability of a polytope X is

$$\begin{aligned} S(X) &= \int_{(i,j)\in X} f(i,j) \\ &= \int_{i\in X_p} I_p(i) f_p(i) \times \int_{j\in X_a} I_a(j) f_a(j) \end{aligned}$$

Recall that the preferences of agents $p_2, ..., p_n$ are dependent on agent p_1 and the preferences of agents $a_2, ..., a_n$ are dependent on a_1. Because the generating process and the preferences are conditionally independently and identically distributed, the probability of a profile $x = [x_1, x_2, ..., x_n]$ is equal to

$$f_p(x) = P(x_1) \prod_{i=2}^{n} P(x_i|x_1)$$

This can then be substituted into the equation for $S(X)$.

However, this method is much more computationally expensive, as the probability of each preference profile must be individually calculated. And in the cases we tested, the direction of comparison does not change even if this method is used. Thus, in the experiments conducted, only the method of profile enumeration is used.

Chapter 6

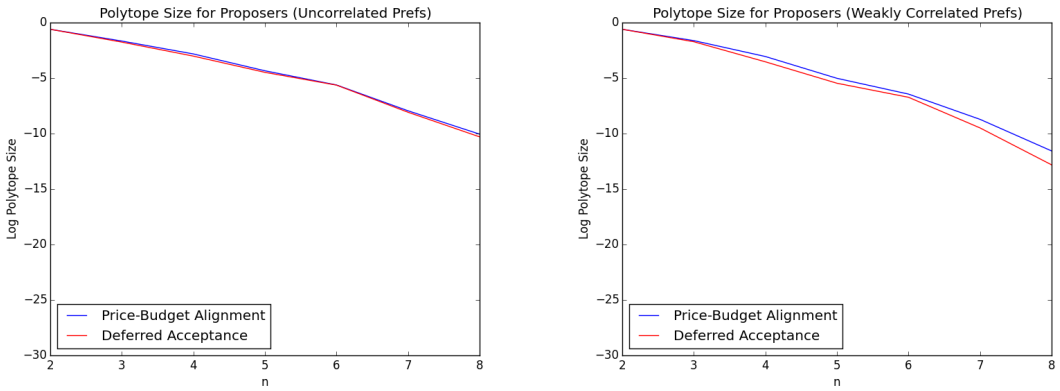
Experimental Results

In this chapter, the performance of price-budget alignment is compared with minimax regret and deferred acceptance, as implemented with incremental preference elicitation. For each measurement, markets with up to 8 agents on each side, over 100 trials with different preference profiles, are tested. The limitations in the size of the economies tested is due to the computational complexity of enumerating the polytope — all three algorithms can be run on economies of large scale efficiently. The comparisons are split up pairwise into price-budget alignment versus deferred acceptance and price-budget alignment versus minimax regret.

6.1 Comparison with Deferred Acceptance

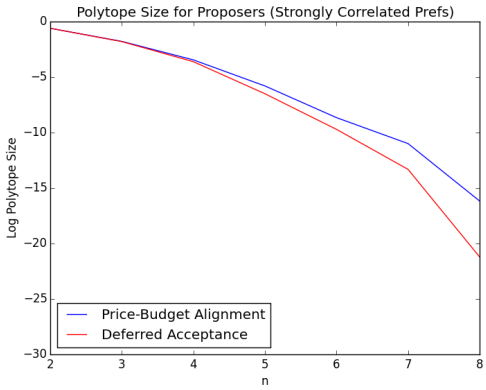
6.1.1 Size of Polytope for Proposers

As can be seen in Figures 6.1a, b, and c, the size of the polytope for proposers was larger under price-budget alignment than under deferred acceptance for all levels of correlation in preferences. This implies price-budget alignment requires less information from proposers than deferred acceptance does. In addition, Figures 6.1b and c show that for weakly and strongly correlated preferences, as the size of the economy increases, the gap between the two mechanisms widens. Furthermore, as the correlation between preferences increases, the gap also widens, which can be seen when comparing between Figures 6.1a, b, and c.



(a) Uncorrelated Preferences

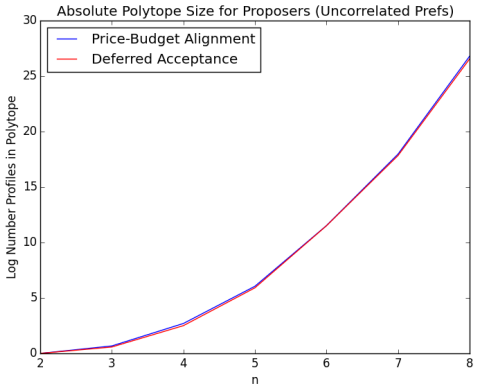
(b) Weakly Correlated Preferences



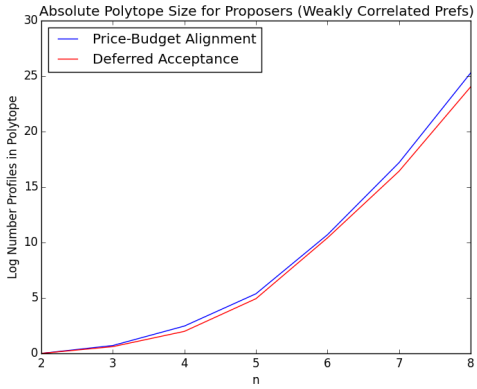
(c) Strongly Correlated Preferences

Figure 6.1: Comparison of Proposer Polytope Size (DA)

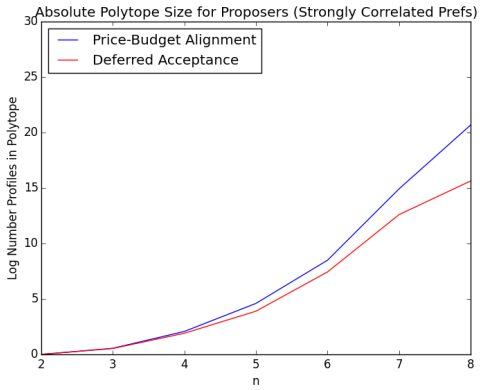
In Figures 6.2a, b, and c, the number of profiles in the polytope, rather than the proportion, is compared between price-budget alignment and deferred acceptance. We can see that there are more possible profiles in the polytope under price-budget alignment than under deferred acceptance. Similarly, the gap between the two mechanisms widens as the size of the economy increases for weak and strong correlation. And as we compare between Figures 6.2a, b, and c, we can again see that the gap also widens as the level of correlation increases.



(a) Uncorrelated Preferences



(b) Weakly Correlated Preferences



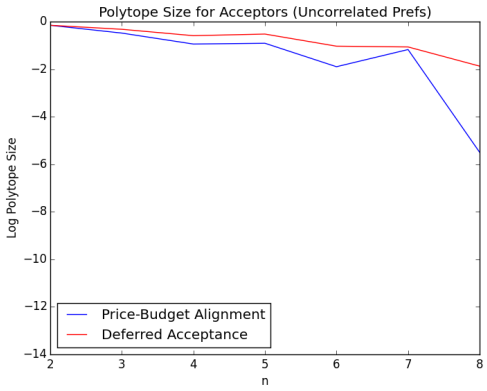
(c) Strongly Correlated Preferences

Figure 6.2: Comparison of Absolute Proposer Polytope Size (DA)

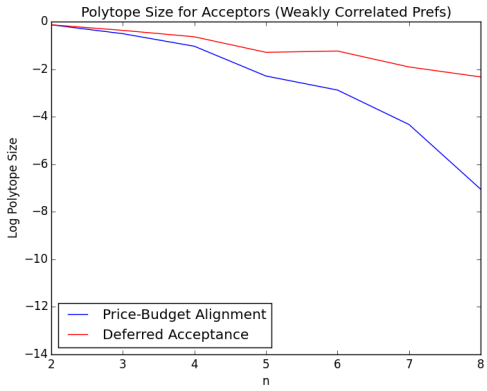
The intuition for this gap is that under price-budget alignment, the prices adjust such that proposers cannot afford acceptors they might have proposed to, and been rejected by, under deferred acceptance. For example, if an acceptor a is over demanded in the first round, the price of a will increase, making her unaffordable to proposers less appealing than her most preferred proposal. Unmatched proposers for whom a is the second choice have the opportunity to proposer to a in deferred acceptance, but in price-budget alignment, may not be able to afford a. In particular in larger economies, this effect is magnified - in each round when the price for an acceptor increases, more and more agents are unable to afford that acceptor. This effect also causes the gap to widen as correlation increases. As the level of correlation increases, proposers are more likely to propose to the same set of acceptors under deferred acceptance, but would not be able to do so under price-budget alignment.

6.1.2 Size of Polytope for Acceptors

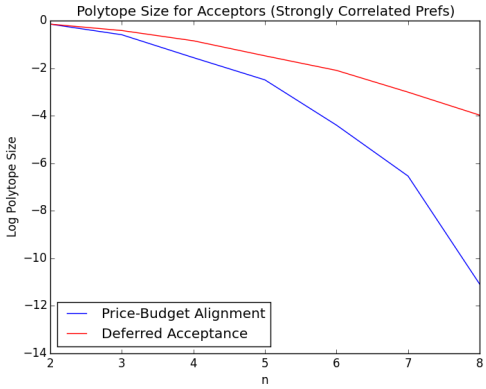
As can be seen in Figures 6.3a, b, and c, the size of the polytope for acceptors was smaller under price-budget alignment than under deferred acceptance. We can see that the gap between deferred acceptance and price-budget alignment widens under all levels of correlation as the size of the market increases. In addition, the gap widens as the level of correlation increases, as can be seen when comparing between Figures 6.3a, b, c.



(a) Uncorrelated Preferences



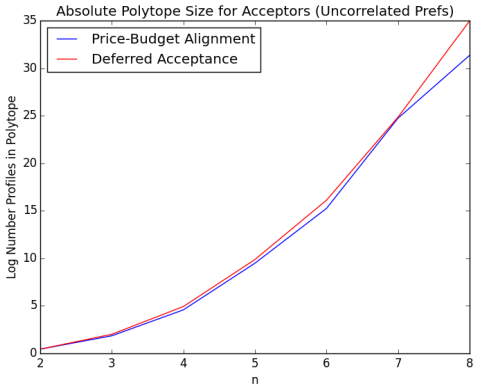
(b) Weakly Correlated Preferences



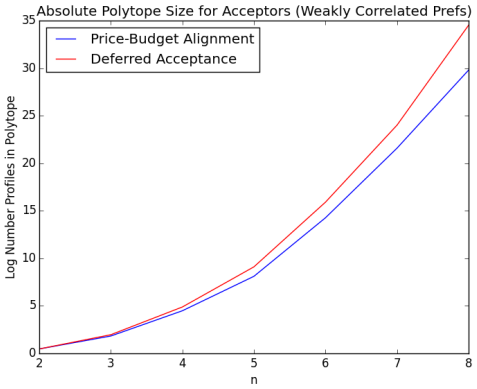
(c) Strongly Correlated Preferences

Figure 6.3: Comparison of Acceptor Polytope Size (DA)

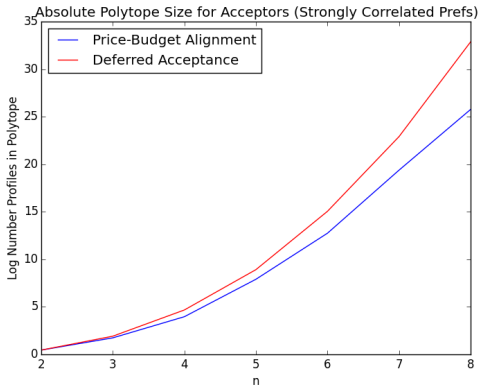
In Figures 6.4a, b, c, the number of profiles in the polytope, rather than the proportion, is compared between price-budget alignment and deferred acceptance. We can see that there are more possible profiles in the polytope under deferred acceptance than under price-budget alignment. A similar effect in the size of the gap with relation to level of correlation and size of the market can be seen when comparing the two mechanisms.



(a) Uncorrelated Preferences



(b) Weakly Correlated Preferences



(c) Strongly Correlated Preferences

Figure 6.4: Comparison of Absolute Acceptor Polytope Size (DA)

The intuition here is actually quite similar to the intuition for why the opposite gap exists for proposers. As mentioned, in price-budget alignment, acceptors who are queried are asked to give information on all proposers, even ones that might not have ever proposed. And in particular, as the size of the economy grows, the number of proposers grows, so acceptors are giving more information that deferred acceptance would not elicit.

Through conducting these experiments, it can seen that both deferred acceptance and price-budget alignment require far more information from proposers than acceptors. This makes any marginal increase in the level of information elicited from proposers significant, as proposers are forced to pin down their preferences at a higher and higher level of accuracy. For acceptors, both algorithms perform generally well, requiring relatively little information from agents, in particular when considering uncorrelated or weakly correlated

preferences. Thus, a marginal increase in information elicited from proposers is of more significance than a marginal increase in information elicited from acceptors.

6.1.3 Iterations

In Figure 6.5, we can see that the number of iterations that each algorithm needs to compute a stable matching was also measured, and on average for each size, price-budget alignment required less iterations than deferred acceptance. This average is taken across all levels of correlation in preferences.

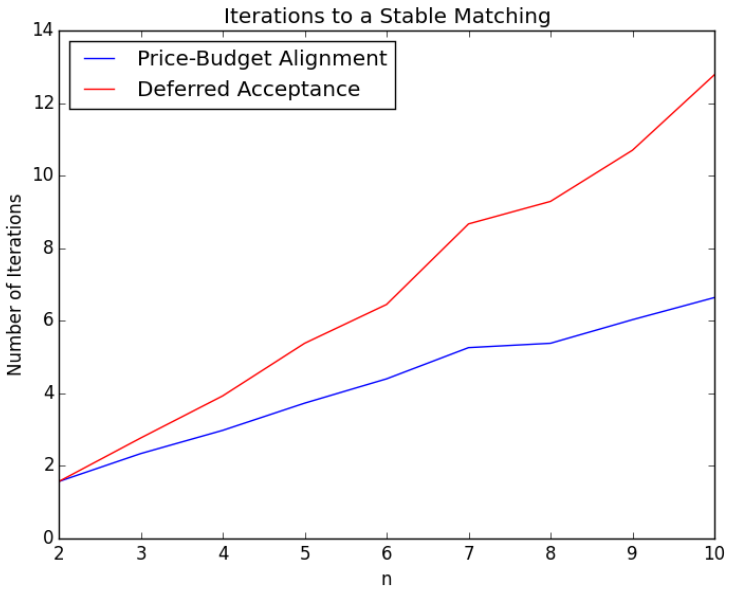


Figure 6.5: Comparison of Iterations Required for Stable Matching (DA)

The intuition for this is again similar to the intuition for the better proposer performance. Because the prices adjust such that proposers cannot afford acceptors they might have proposed to, and been rejected by, under deferred acceptance, fewer proposals are needed to arrive at a stable matching. Thus, fewer iterations are required for the mechanism to converge. This ties in well with the polytope comparisons — because more information is required from acceptors each round under price-budget alignment, proposers require fewer iterations to achieve a stable matching, and less information is thus required from proposers.

6.2 Comparison with Minimax Optimal Matching

First, we note that in recording the number of pairwise comparisons that are asked of agents, the minimax regret algorithm does not make it clear how much information is collected in a halving query. Drummond and Boutilier [2] measure information collection through number of halving queries, rather than information collected. It is clear that the number of pairwise comparisons needed to divide a block into a more preferred half and a less preferred half is less than the number needed to completely sort the block. Thus, to be generous to the minimax algorithm, we use the underlying true ranking to sort the block, then only record the pairwise comparisons that emerge from having an unsorted better and lesser half. For example, if a preference block is

$$B = [1, 2, 4, 5]$$

and the true preference order is

$$\succ = [4, 1, 5, 2]$$

then we use the true order to split B into

$$B' = [[1, 4], [2, 5]]$$

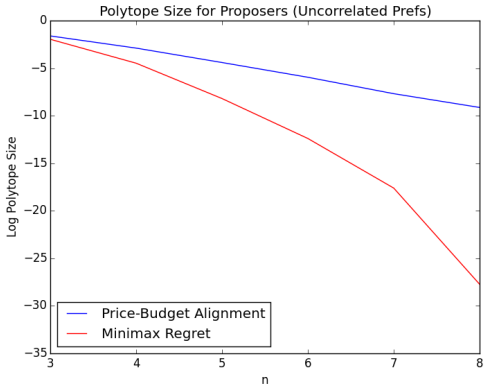
and record the pairwise queries to be

$$I = [(4, 5), (4, 2), (1, 5), (1, 2)]$$

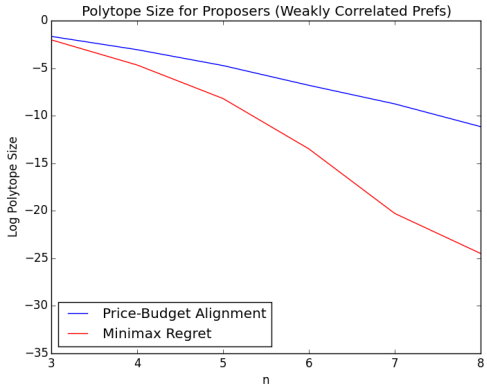
Note that this is a lower bound on the true number of queries and information necessary to split B into B'.

6.2.1 Size of Polytope for Proposers

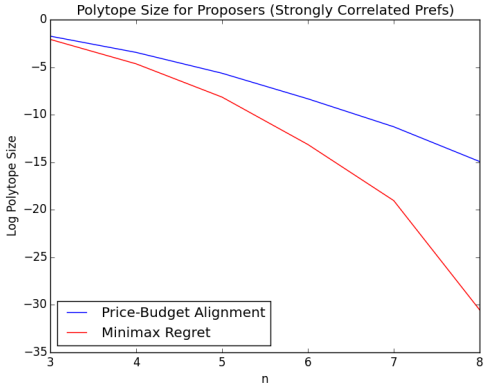
As can be seen in Figures 6.6a, b, and c, the size of the polytope for proposers is larger under price-budget alignment than under minimax regret, and the gap widens as the size of the economy increases for all levels of correlation. In addition, the gap remains about the same even as the level of correlation increases, as can be seen when comparing between Figures 6.6a, b, and c. We also note that both algorithms require a high level of information from proposers.



(a) Uncorrelated Preferences



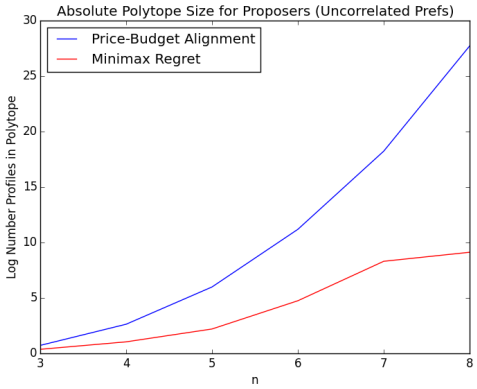
(b) Weakly Correlated Preferences



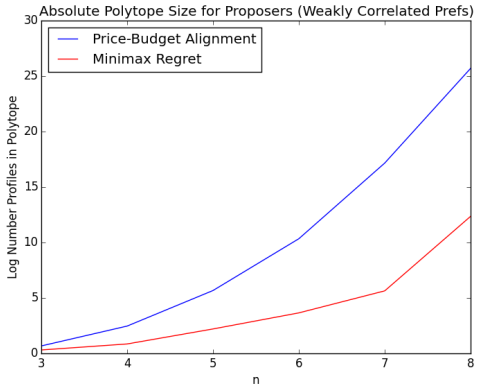
(c) Strongly Correlated Preferences

Figure 6.6: Comparison of Proposer Polytope Size (MMR)

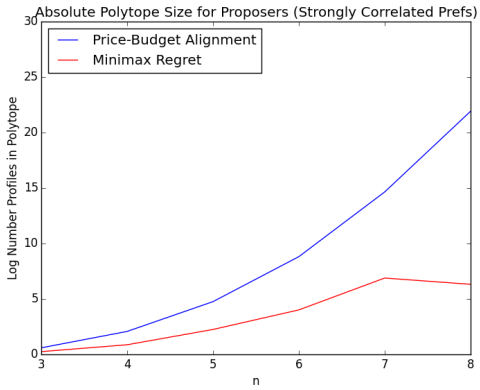
In Figure 6.7, the number of profiles in the proposer polytope, rather than the proportion, is compared between price-budget alignment and minimax regret. We can see that there are more possible profiles in the polytope under price-budget alignment than under minimax regret. And, again we note that as the size of the economy increases, regardless of correlation level, the gap widens. In addition, the size of the gap remains about the same regardless of the level of correlation in preferences, as can be seen when comparing between Figures 6.7a, b, and c.



(a) Uncorrelated Preferences



(b) Weakly Correlated Preferences



(c) Strongly Correlated Preferences

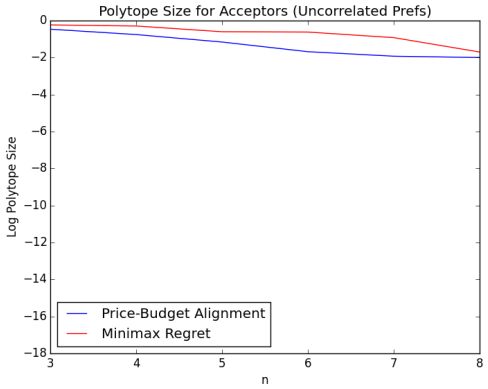
Figure 6.7: Comparison of Proposer Polytope Size (MMR)

We first note that each proposer interview in minimax regret elicits more information than proposer interviews in price-budget alignment. The proposer polytope is constricted less after asking a proposer to pick her most preferred acceptor out of a subset of all acceptors than after asking a proposer to split a block of acceptors into more and less preferred halves. With fewer iterations, as Figure 6.10 shows, this results in a larger proposer polytope under price-budget alignment versus minimax regret.

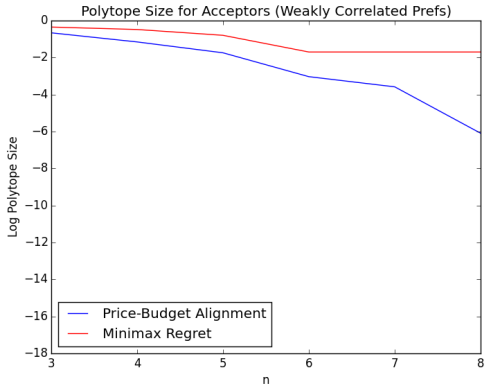
6.2.2 Size of Polytope for Acceptors

For acceptors, we can see via Figures 6.8a, and b, that with uncorrelated and weakly correlated preferences, the size of the polytope for acceptors is larger under minimax regret. However, as Figure 6.8c shows, with strongly correlated preferences, this changes at $n = 8$,

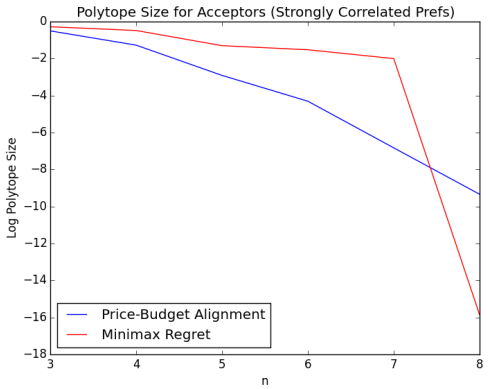
with a larger acceptor polytope under price-budget alignment. To further examine this change, we performed the same experiment with perfectly correlated preferences and, as can be seen in Figure 6.8d, found that for $n > 4$, the acceptor polytope is larger under price-budget alignment. Also of note, under both mechanisms, the level of information collected for acceptors is of smaller magnitude than that of proposers.



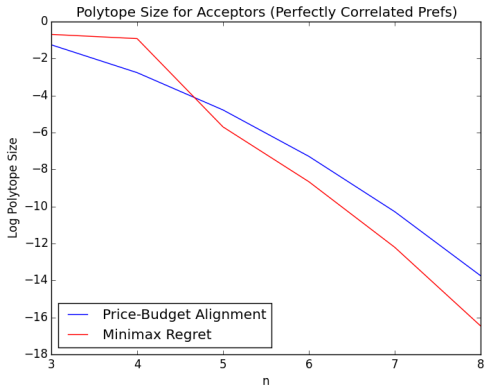
(a) Uncorrelated Preferences



(b) Weakly Correlated Preferences



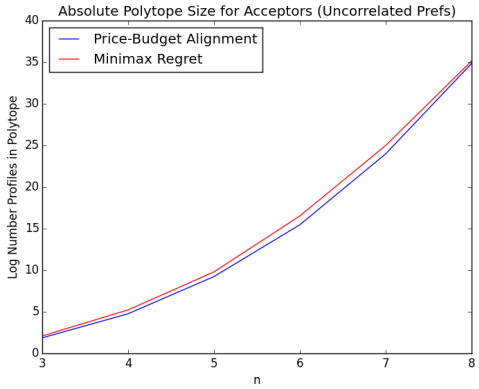
(c) Strongly Correlated Preferences



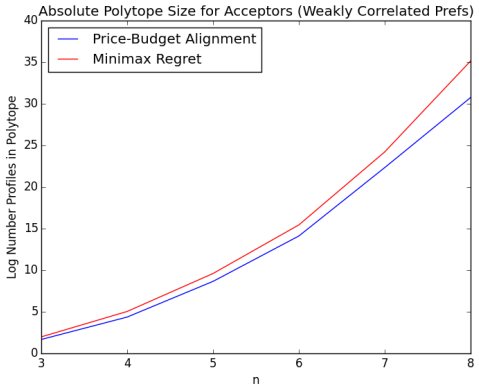
(d) Perfectly Correlated Preferences

Figure 6.8: Comparison of Acceptor Polytope Size (MMR)

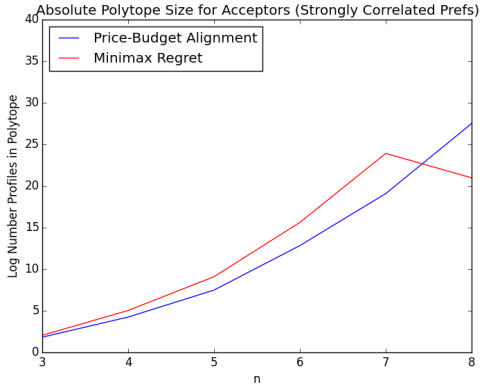
In Figures 6.9a, b, c, and d, the number of profiles in the proposer polytope, rather than the proportion, is compared between price-budget alignment and minimax regret. Similarly we note that the number of profiles in the polytope is greater under minimax regret with uncorrelated and weakly correlated preferences. However, price-budget alignment results in a larger number of profiles in the polytope at $n = 8$ for strongly correlated preferences and at $n = 5$ for perfectly correlated preferences.



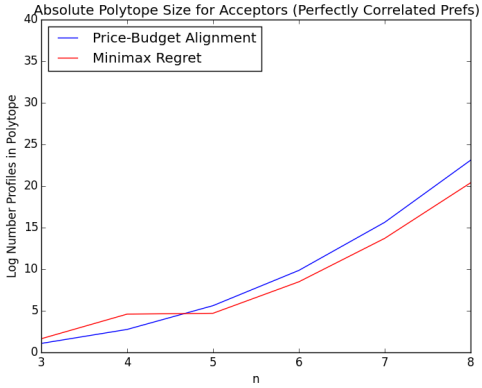
(a) Uncorrelated Preferences



(b) Weakly Correlated Preferences



(c) Strongly Correlated Preferences



(d) Perfectly Correlated Preferences

Figure 6.9: Comparison of Acceptor Polytope Size (MMR)

First, we note that for n small, there is a relatively small number of possible matchings, and thus few matchings with the minimum maximum regret value. So, it is fairly likely a stable matching, rather than an unstable matching, is picked each round, allowing the algorithm to require relatively few agent interviews. As n grows, the number of matchings with the minimum maximum regret value each round grows as well, leading to a higher probability that an unstable matching is chosen each round. How quickly the matching stabilizes depends on the level of information elicited.

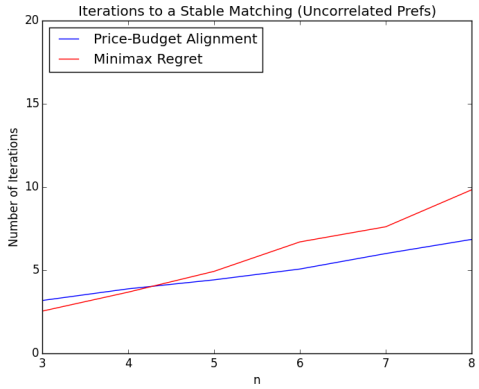
Now, note that each acceptor interview in minimax regret elicits less information than acceptor interviews in price-budget alignment. The acceptor polytope is constricted more after asking an acceptor to pivot all proposers around one agent than after asking an acceptor to split a block of proposers into more and less preferred halves. With a larger polytope,

the possible matchings with the same minimum maximum regret value increases. When n is small, this is not a problem, as mentioned earlier — a stable matching will be found relatively quickly, and thus less information is required under minimax regret than under price-budget alignment. However, as n grows, the number of interviews required to constrict the polytope enough for a stable matching surpasses that of price-budget alignment, thus requiring more total information from acceptors.

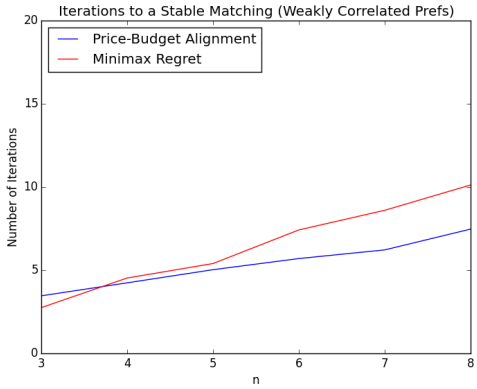
Similarly, as the correlation in agent preferences increases, the less information collected from acceptors, the more interviews necessary to reach a stable matching. With more acceptor interviews, the total information collected from acceptors increases, shrinking the size of the acceptor polytope.

6.2.3 Iterations

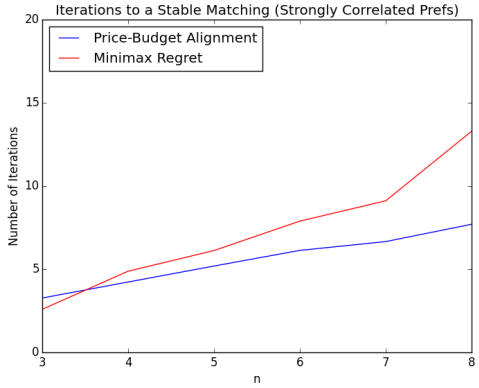
As Figures 6.10a, b, c, d show, the number of iterations needed to achieve a stable matching is initially smaller for minimax, but as n grows, begins to swap such such that price-budget alignment requires fewer iterations. In Figure 6.10a, we see that the change begins at $n = 5$ when using uncorrelated preferences. And in Figures 6.10b, c, and d, we see that the change begins at $n = 4$ when using correlated preferences.



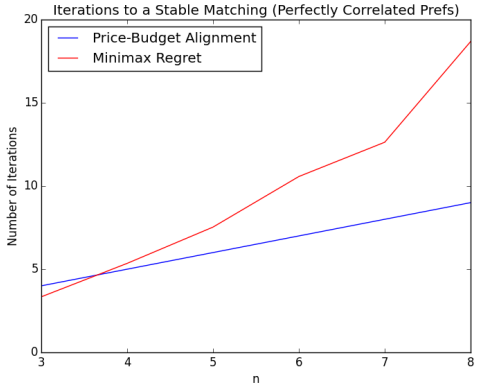
(a) Uncorrelated Preferences



(b) Weakly Correlated Preferences



(c) Strongly Correlated Preferences



(d) Perfectly Correlated Preferences

Figure 6.10: Comparison of Iterations Required for Stable Matching(MMR)

The intuition here follows from the intuition for the acceptor polytope comparison: as n grows, more information is needed to use minimax regret to approximate a stable matching. This increases the number of interviews needed, which in turn increases the iterations necessary to reach a stable matching. Furthermore, as can be seen by comparing Figures 6.10a, b, c, and d, the gap between minimax and price-budget alignment widens as the correlation in preferences increases. This also follows from our analysis of the acceptor polytope sizes, in particular, as correlation increases, the gaps in Figures 6.10 widen, and the points at which price-budget alignment requires less information than minimax regret in Figures 6.9 happen earlier.

We note that Drummond and Boutilier [2] compare minimax regret with Gale-Shapley for economies of size $n = 20$ and $n = 250$. For $n = 250$, they found that minimax regret

requires less queries from both proposers and acceptors than Gale-Shapley under correlated preferences, and requires more from both sides under uncorrelated preferences. For $n = 20$, they found that minimax regret requires about one more query per acceptor than Gale-Shapley and about six fewer queries per proposer under both correlated and uncorrelated preferences. However, we found that for sizes $n = 3$ to $n = 8$, the proposer polytope under minimax regret is smaller for proposers and about the same size for acceptors in comparison to Gale-Shapley. While the economy sizes are not exactly the same, if the trends continue, this implies that the additional, marginal query per acceptor does not constrict the polytope by too much, and while fewer queries from proposers are needed, they require more information. We note that more information is elicited per proposer query by minimax regret than by price-budget alignment, and the proposer queries for Gale-Shapley and price-budget alignment are similar. This implies that more information is elicited per proposer query by minimax regret than by Gale-Shapley, which matches our above analysis.

Chapter 7

Conclusion

7.1 Implications

Through the comparative analysis, we find that the price-budget alignment algorithm requires less information from proposers than both deferred acceptance and minimax regret, but more information from acceptors. When considering some of the real-life applications, we note that there are settings where the preferences of the accepting side are clear-cut and easy to describe, whereas the preferences of the proposing side are more difficult to describe. In these settings, it may be more effective to use the price-budget alignment algorithm such that less information is required from the proposers and more information is required from the acceptors.

For example, in resident matching, the proposer is the student who applies to various schools, and the acceptors are schools. For a student, ranking schools may be more difficult considering the many different factors such as academics, research programs, student-life, extracurriculars, etc. But for schools, students may be more easily ranked with quantitative measurements for GPA and MCAT scores.

7.2 Future Work

Although the attempts at understanding the stable marriage primal in a way that supports a primal-dual algorithm were not successful, it would be interesting to consider the construction of an alternative linear formulation that captures the blocking constraint in the objective, rather than in the constraints. Further progress is also likely possible with

the iterative, elicitation method described as well. While the algorithm requires less information from proposers, ideally, less information would also be required from acceptors as well. Different methods of utilizing the prior probabilities for correlated preferences is a possible way to improve the elicitation mechanism.

Appendix A

Matching Mechanisms

A.1 Full Minimax Regret Algorithm

Algorithm 3 Minimax Regret

```
1: procedure MMR(E, A)
2:     P ← ∅
3:     loop
4:         choose a completion ≻∈ C(P)
5:         use Gale-Shapley to find a stable matching μ using ≻
6:         if MMR(μ, P) ≤ τ then
7:             return μ
8:         for q ∈ RI(μ) s.t. q not queried this round do
9:             if r and μ(q) are in the same block B of P_q for some r ∈ BP(q) then
10:                Ask q to split B
11:            else
12:                for r ∈ BP(q) s.t. r not queried this round do
13:                    if q and μ(r) are in the same block B of P_r then
14:                        Ask r to split B
15:        if no preferences were updated this round then
16:            for P_j ∈ P do
17:                Ask j to split j's largest block in P_j
```

A.2 Full Lazy Gale Shapley Algorithm

Algorithm 4 Lazy Gale Shapley

procedure LGS($E, A, c_{E,A}$)
  $l \leftarrow 0$
  $\mu(e_i) \leftarrow \emptyset, \mu(a_j) \leftarrow \emptyset, \forall e_i \in E, \forall a_j \in A$
  $X_{e_i} \leftarrow \emptyset, \forall e_i \in E$
  while at least one proposer is not matched and has possible achievable matches **do**
    for $e_i \in E$ **do**
      if $\mu(e_i) = \emptyset$ **then**
        $P_{e_i} \leftarrow$ set of achievable acceptors in the highest-ranked equivalence class of e_i among those with his achievable applicants, whom he has not interviewed yet
    $E^{\mu,l} \leftarrow$ set of unmatched e_d in equiv. class l of the acceptors with $P_{e_d} \neq \emptyset$
    while $E^{\mu,l} = \emptyset$ **do**
      $l \leftarrow l + 1$
      $E^{\mu,l} \leftarrow$ the set of unmatched proposers e_d in equivalence class l of the acceptors with nonempty P_{e_d}
    $e_i \leftarrow$ an arbitrary proposer in $E^{\mu,l}$
    for $a_j \in P_{e_i}$ **do**
      e_i interview a_j
    $X_{e_i} = P_{e_i}$
    $P_{e_i} \leftarrow \emptyset$
    while exists an unmatched employer e_i with $X_{e_i} = \emptyset$ **do**
      for $e_i \in E$ **do**
        if $\mu(e_i) = \emptyset$ and $X_{e_i} \neq \emptyset$ **then**
          e_i proposes to the achievable acceptor it prefers the most.
          Remove that acceptor from X_{e_i}
      Update μ such that each acceptor who has received one or more proposals tentatively accepts the proposer he most prefers and rejects the rest.
      X_{e_i} is updated for all $e_i \in E$ such that each tentatively matched acceptor is no more achievable to those proposers that are in the equivalence classes ranked lower than the one $\mu(a_j$ belongs to - he is removed from the lists of such proposers.
  return μ

Appendix B

Linear Formulations

B.1 Single-Sided Assignment

Assume we have a set A of n agents and a set T of n items, where item j has value v_{ij} to agent i and x_{ij} is a binary representing allocation of item j to agent i. Each agent can only be allocated one item and each item can only be allocated to one agent. The objective of both sides of the economy is, again, to maximize utility.

B.1.1 LP and Dual

Our primal is as follows:

$$\begin{aligned} \max & \sum_{i \in A} \sum_{j \in T} x_{ij} v_{ij} \\ \text{s.t.} & \sum_{j \in T} x_{ij} \leq 1, \quad \forall i \in A \\ & \sum_{i \in A} x_{ij} \leq 1, \quad \forall j \in T \\ & x_{ij} \geq 0, \quad \forall (i,j) \in A \times T \end{aligned}$$

The dual is then:

$$\begin{aligned} \min & \sum_{j \in T} p_j + \sum_{i \in A} \pi_i \\ \text{s.t. } & p_j + \pi_i \geq v_{ij} \quad \forall (i,j) \in A \times T \\ & p_j \geq 0, \quad \forall j \in T \\ & \pi_i \geq 0, \quad \forall i \in A \end{aligned}$$

We note that p has a natural interpretation as price of item j and π has a natural interpretation as profit of agent i.

B.1.2 Complementary Slackness

We note the following implications of the complementary slackness theorem.

1. $x_{ij} > 0 \implies \pi_i + p_j = v_{ij}$
2. $\pi_i > 0 \implies \sum_{j \in T} x_{ij} = 1$
3. $p_j > 0 \implies \sum_{i \in B} x_{ij} = 1$

By the first implication, if item j is allocated to agent i, then the profit of agent i is equal to the difference between the value gained and the price paid. By the second implication, if agent i has non-zero profit, an item must have been allocated to him. And by the third implication, if item j has non-zero price, it must have been allocated to some agent. Intuitively, these all make sense, and we can use them to formulate an iterative algorithm.

B.1.3 Iterative Algorithm

Let C be our complementary slackness conditions. Then we have the following iterative algorithm:

Algorithm 5 Single Sided Assignment

```
1: procedure MATCH(A, T, V)
2:     p_j ← 0 for all j ∈ T
3:     optimal ← false
4:     while not optimal do
5:         for i ∈ A do
6:             π_j ← max_{j∈T} v_ij − p_j
7:             match[i] = arg(max_{j∈T} v_ij − p_j)
8:         if match satisfies C then
9:             optimal ← true
10:        else
11:            for j ∈ T s.t. j is allocated to more than one agent do
12:                p_j+ = 1
```

B.2 Two-Sided Assignment

To further add complexity, and approach the marriage problem, we consider the two-sided assignment problem. There are two finite, disjoint sets of size n representing agents. Let the sets be denoted P, A. The parameters are $a_{ij} : (i, j) \in P \times A$ where α_{ij} is the combined utility of a match between agent i and agent j. Further x_{ij} is a binary representing whether or not agent i and agent j are matched.

B.2.1 LP and Dual

The primal is as follows:

$$\begin{aligned} \max_{(i,j)\in M\times W} \quad & \alpha_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j\in W} x_{ij} \leq 1, \quad \forall i \in M \\ & \sum_{i\in M} x_{ij} \leq 1, \quad \forall j \in W \\ & x_{ij} \geq 0, \quad \forall (i,j) \in M \times W \end{aligned}$$

And the dual is as follows:

$$\begin{aligned} \min \sum_{i \in M} u_i + \sum_{j \in W} v_j \\ \text{s.t. } u_i + v_j \geq \alpha_{ij}, \quad \forall (i,j) \in M \times W \\ u_i, v_j \geq 0, \quad \forall i \in M, j \in W \end{aligned}$$

We note that u_i corresponds to the utility of $i \in M$ and v_j corresponds to the utility of $j \in W$.

B.2.2 Complementary Slackness

We note the following implications of the complementary slackness theorem.

1. $u_i > 0 \implies \sum_{j \in W} x_{ij} = 1$
2. $v_j > 0 \implies \sum_{i \in M} x_{ij} = 1$
3. $x_{ij} > 0 \implies u_i + v_j = \alpha_{ij}$

The first two indicates that a non-zero utility for an agent implies the agent is matched to someone. The third indicates that a match between agent i and j implies their utility is equal to the value of the match.

B.2.3 Finding Prices

In the auction and single-assignment problems, we were able to find prices from the dual. However, here, in the two-sided assignment problem, the dual variables represent utility, rather than prices. We can introduce redundancy into the formulation to find prices. Our primal is then as follows:

$$\begin{aligned} \max \sum_{(i,j) \in M \times W} a_{ij} x_{ij} \\ \text{s.t. } \sum_{j \in W} x_{ij} \leq 1, \quad \forall i \in M \\ \sum_{i \in M} x_{ij} \leq 1, \quad \forall j \in W \\ x_{ij} \leq 1, \quad \forall (i,j) \in M \times W \\ x_{ij} \geq 0, \quad \forall (i,j) \in M \times W \end{aligned}$$

And the dual is as follows:

$$\begin{aligned} \min \sum_{i \in M} u_i + \sum_{j \in W} v_j + \sum_{(i,j) \in M \times W} p_{ij} \\ \text{s.t. } u_i + v_j + p_{ij} \geq a_{ij}, \quad \forall (i,j) \in M \times W \\ u_i, v_j \geq 0, \quad \forall i \in M, j \in W \end{aligned}$$

We can interpret the dual variables similarly to above, with the added variable p_{ij} as the cost of matching agent i to agent j.

B.2.4 Complementary Slackness with Prices

We note the following implications of the complementary slackness theorem

1. $u_i > 0 \implies \sum_{j \in W} x_{ij} = 1$
2. $v_j > 0 \implies \sum_{i \in M} x_{ij} = 1$
3. $p_{ij} > 0 \implies x_{ij} = 1$
4. $x_{ij} > 0 \implies u_i + v_j = \alpha_{ij} - p_{ij}$

The interpretation of the first two implications is that if an agent $i \in M$ has a non-zero utility, the agent must be matched. The third implication means that if the price of forming a match between agents i, j is non-zero, they must be matched. And the last implication means that if agents i, j are matched, then their total utility is equal to the value of the match minus the cost of the match. The dual thus provides both payoff and price variables which make an iterative primal-dual mechanism possible.

Bibliography

[1] Atila Abdulkadiroglu, Parag A. Pathak, and Alvin E. Roth. The New York High School Match. *American Economic Review*, 95(2), 2005.

[2] Joanna Drummond and Craig Boutilier. Elicitation and approximately stable matching with partial preferences. *IJCAI International Joint Conference on Artificial Intelligence*, pages 97–105, 2013.

[3] Joanna Drummond and Craig Boutilier. Preference Elicitation and Interview Minimization in Stable Matchings (working paper). 2016.

[4] B. Edelman, M. Ostrovsky, and Schwarz M. Internet advertising and the generalized price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97:242–259, 2007.

[5] D Gale and LS Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 1962.

[6] D.E. Knuth. Stable Marriage and its Relation to Other Combinatorial Problems, CRM Proceedings and Lecture Notes. *American Mathematical Society*, 10, 1997.

[7] David C. Parkes. Iterative Combinatorial Auctions: Achieving Economic And Computational Efficienty. *PhD Theses*, 2001.

[8] David C Parkes and Lyle H. Ungar. Iterative Combinatorial Auctions: Theory and Practice. *In Proc. 17th National conference on Artificial Intelligence (AAAI-00)*, pages 74–81, 2000.

[9] Baharak Rastegari, Anne Condon, Nicole Immorlica, and Kevin Leyton-Brown. Two-sided matching with partial information. *Proceedings of the fourteenth ACM conference on Electronic commerce - EC '13*, 1(212):733, 2013.

[10] Alvin E. Roth and Elliott Peranson. The Redesign of the Matching Market for American Physicians: Some Engineering Aspects of Economic Design. *American Economic Review*, 89(4):748–780, 1999.

[11] Alvin E. Roth, Uriel G. Rothblum, and John H Vande Vate. Stable Matchings, Optimal Assignments, and Linear Programming, 1993.

[12] C.-P. Teo, J. Sethuraman, and W.-P. Tan. Gale-Shapley stable marriage problem revisited: strategic issues and applications. *Proc. of IPCO '99: the 7th Conference on Integer Programming and Combinatorial Optimisation*, pages 429–438, 1999.

[13] John H. Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(DMC):147–153, 1989.

[14] Y.L. Varol and D. Rotem. An algorithm to generate all topological sorting arrangements. *The Computer Journal*, 24:83–84, 1979.

[15] Rakesh V Vohra. Stable matchings and linear programming. *Current Science(Bangalore)*, 103(9):1051–1055, 2012.