DISKRETE MATHEMATIK UND OPTIMIERUNG

Winfried Hochstättler, Robert Nickel and David Schiess:

Mixed Matching Markets

Technical Report feu-dmo010.08
Contact: winfried.hochsta ettler@fernuni-hagen.de, robert.nickel@fernuni-hagen.de, david.schiess@unisg.ch

FernUniversität in Hagen
Lehrgebiet Mathematik
Lehrstuhl für Diskrete Mathematik und Optimierung
D – 58084 Hagen

2000 Mathematics Subject Classification: 05D1591A12,91B26
Keywords: stable marriage, assignment game, core

Mixed Matching Markets

Winfried Hochstättler
Department of Mathematics
FernUniversität in Hagen
D-58084 Hagen

Robert Nickel
Department of Mathematics
FernUniversität in Hagen
D-58084 Hagen

David Schiess
Department of Mathematics and Statistics
University of St. Gallen
CH-9000 St. Gallen

January 24, 2008

Abstract

We introduce a new model for two-sided markets that generalizes stable marriages as well as assignment games. Our model is a further generalization of the model introduced by Eriksson and Karlander [2]. We prove that the core of our model is always non-empty by providing an algorithm that determines a stable solution in $\mathcal{O}(n^4)$.

1 Introduction

The stable marriage problem introduced by Gale and Shapley [3] is quite well known to scientists from different fields such as game theory, economics, computer science, and combinatorial optimization. There are at least three monographs, by Knuth [6], Gusfield and Irving [4] and Roth and Sotomayor [8] devoted to it. The problem reads as follows: given two disjoint groups of players (men-women or workers-firms etc.), where each player is endowed with a preference list on the other group, the objective is to match the players from one group to players from the other group such that there is no pair which is not matched but prefers each other over their partners. The existence of such a *stable matching* is proved algorithmically using the so called "men propose – women dispose"-algorithm given in [3].

In their monograph [8] Roth and Sotomayor observed that the set of stable solutions from another game on bipartite matching, namely the assignment game [11], has several structural similarities with the set of stable matchings. They challenged the readers to find a unifying theory for the two games. In the assignment game we are given a weighted bipartite graph. A solution consists of a matching and an allocation of its weight to the players. A solution is stable if no pair gets allocated less than the weight of its connecting edge. Shapley and Shubik [11] observed that this condition is identical to the dual constraints of the linear programming model for weighted bipartite matching, thus the dual variables in an optimal solution coincide with the stable allocations. Roth and Sotomayor [9] themselves presented a first model unifying stable marriage and the assignment game and showed that its set of stable solutions, if it is non-empty, has the desired structural properties. Eriksson and Karlander [2] modified this model and gave an algorithmic proof of the existence of a stable solution. For the classical special cases, their algorithm coincides with "men propose – women dispose", respectively with the "exact" auction procedure of [1]. As implemented, this algorithm is not polynomial time but pseudopolynomial. A careful analysis of the algorithm (see [5]), though, reveals that a proper implementation solves the problem in $\mathcal{O}(n^4)$.

The purpose of the present paper is to present a further natural generalization of these games. While in the model of Eriksson and Karlander the players are partitioned into rigid and flexible players and the distribution of the value of an edge is flexible only if both players are flexible, in our model each possible pair of players may choose whether it closes a flexible or a rigid contract. This mirrors the situation on labor markets nowadays. In addition the input data becomes simpler and the assignment game is no longer interpreted as "a stable marriage game with side payments" (see [2]).

We generalize the algorithm from [5] to the new model and show that it computes a stable solution in $\mathcal{O}(n^4)$.

In the next section we introduce the model and discuss its special cases. In Section 3 we present our algorithm. Sections 4 and 5 are devoted to the correctness proof of the algorithm and its running time analysis. We assume some familiarity with bipartite matching and combinatorial optimization. Our notation should be fairly standard.

2 The Model

Given three non-negative real square matrices $A = (a_{ij}), B = (b_{ij}), C = (c_{ij}) \in \mathbb{R}_+^{n\times n}$ an *outcome* is a *matching permutation* $\sigma \in S_n$, a *flexibility map* $f : \{1, \dots, n\} \to \{0, 1\}$ and two *payoff functions* $u : \{1, \dots, n\} \to \mathbb{R}_+$, $v : \{1, \dots, n\} \to \mathbb{R}_+$ such that

$$\begin{aligned} \forall 1 \le i \le n: \quad & (\text{ if } f(i) = 1 \text{ then } \quad u_i + v_{\sigma(i)} = c_{i\sigma(i)}) \\ \text{and} \quad & (\text{ if } f(i) = 0 \text{ then } \quad u_i = a_{i\sigma(i)} \text{ and } v_{\sigma(i)} = b_{i\sigma(i)}). \end{aligned}$$

The outcome is *stable* if in addition

$$\forall 1 \le i \le n \quad \forall 1 \le j \le n: \quad u_i + v_j \ge c_{ij} \tag{1}$$

$$\forall 1 \le i \le n \quad \forall 1 \le j \le n: \quad u_i \ge a_{ij} \text{ or } v_j \ge b_{ij}. \tag{2}$$

We write an outcome as the quadruple (σ, f, u, v). Since in our algorithm presented later we use an alternating path technique from matching theory we will frequently use P for the index set of the rows and Q for the set of indices of the columns of the matrices. We will identify the data with a complete bipartite graph K_{PQ} which has two copies of each edge. For one of the copies we have the weight function c_{ij} and for the other the pair of weight functions (a_{ij}, b_{ij}). The permutation together with the flexibility function f then corresponds to a perfect matching, where we choose the edge weight $c_{i\sigma(i)}$ if $f(i) = 1$ and $(a_{i\sigma(i)}, b_{i\sigma(i)})$ if $f(i) = 0$.

Remark 1. *In our matching games we assume that the graphs are complete and that both color classes have the same cardinality. This can always be achieved by adding an appropriate number of dummy vertices and corresponding edges of weight zero.*

2.1 Stable Marriages

The input for a stable marriage game consists of a complete bipartite graph $K_{nn} = K_{PQ}$ and for each vertex $i \in P$ resp. $j \in Q$ of a total order $\leq_i$ on the vertices of Q respectively of a total order $\leq_j$ on the vertices of P. We call these total orders *preference lists*. A perfect matching, also called a *marriage*, which we may identify with a permutation $\sigma \in S_n$, is stable, if $\forall i \in P \forall j \in Q : \sigma(i) \geq_i j$ or $\sigma^{-1}(j) \geq_j i$, i.e. if there is no unmatched pair that prefers each other to their matching partners.

Consider our model in the case that $C = 0$ is the zero-matrix and (σ, f, u, v) are a stable outcome. If for some $i \in P$ we have $f(i) = 1$, then $u_i + v_{\sigma(i)} = c_{i\sigma(i)} = 0$ which implies $u_i = v_{\sigma(i)} = 0$ since the payoffs are non-negative. The outcome is stable and hence $0 = u_i \geq a_{i\sigma(i)}$ or $0 = v_{\sigma(i)} \geq b_{i\sigma(i)}$. If only one of $a_{i\sigma(i)}$ and $b_{i\sigma(i)}$ is zero, say $a_{i\sigma(i)} > 0$ then we may change u_i to $a_{i\sigma(i)}$ and set $f(i) = 0$ without changing the stability of the outcome. In addition the total payoff of the game possibly increases. Possibly applying this several times we may assume that f is constantly zero. Now interpreting the numbers a_{ij} for fixed i as priority on the matching partners j by defining $j \geq_i k \iff a_{ij} \geq a_{ik}$ and similarly for the b_{ij} we derive an input example for the stable marriage game and the permutation σ clearly gives a stable marriage.

On the other hand, given an input of a stable marriage game we may represent the preference lists by two strictly positive matrices $A, B \in \mathbb{R}_+^{n \times n}$ and set $C = 0$. Then the stable outcomes of our model are in one-one correspondence with the stable marriages of the given game.

2.2 Assignment Games

In the assignment model we are given a complete bipartite graph $K_{nn} = K_{PQ}$ with non-negative weights $C = (c_{ij})$. A perfect matching $\sigma \in S_n$ together with two payoff functions $u : P \to \mathbb{R}_+$, $v : Q \to \mathbb{R}_+$ such that

$$\forall i \in P : u_i + v_{\sigma(i)} = c_{i\sigma(i)}$$

is called an *outcome*. The outcome is *stable* if additionally

$$\forall i \in P \, \forall j \in Q : u_i + v_j \geq c_{ij},$$

i.e. if there is no unmatched pair that could individually improve by leaving their present partners and forming a new matching edge instead.

If in our model $A = B = 0$ and for a stable outcome σ, f, u, v we have $f(i) = 0$ for some $1 \leq i \leq n$ then necessarily $u_i = v_{\sigma(i)} = 0$. By the stability assumption, $0 = u_i + v_{\sigma(i)} \geq c_{i\sigma(i)} \geq 0$ holds. Hence we also have $c_{i\sigma(i)} = 0$ and modifying f by setting $f(i)$ to 1 maintains stability of the payoff.

Similarly, an input of the assignment game immediately translates into an input of our model and - apart from the degeneracy with edges of weight zero described above - we have a one-one correspondence between stable outcomes in both models.

2.3 The Eriksson-Karlander Model

The model of Eriksson and Karlander was the first generalization of stable matching and the assignment game shown to always admit a stable solution. The input consists of two non-negative square matrices $A, B \in \mathbb{R}_+^{n \times n}$ and a partition $P \dot\cup Q = R \dot\cup F$ of the set of vertices into *flexible* players F and *rigid* players R. We may consider A, B, R as the input data of such a game. An *outcome* is a *matching permutation* $\sigma \in S_n$ and two *payoff functions* $u : P \to \mathbb{R}_+$, $v : Q \to \mathbb{R}_+$ such that

$$\begin{array}{lll} \forall i \in P: & \text{if } \{i, \sigma(i)\} \cap R = \emptyset \text{ then} & u_i + v_{\sigma(i)} = c_{i\sigma(i)} \\ & \text{otherwise} & u_i = a_{i\sigma(i)} \text{ and } v_{\sigma(i)} = b_{i\sigma(i)}. \end{array}$$

The outcome is *stable* if in addition

$$\begin{array}{lll} \forall i \in P \, \forall j \in Q: & \text{if } \{i, \sigma(i)\} \cap R = \emptyset & \text{then } u_i + v_j \geq c_{ij} \\ & \text{otherwise} & u_i \geq a_{ij} \text{ or } v_j \geq b_{ij}. \end{array}$$

Given an instance A, B, R of the Eriksson-Karlander game we define the matrix $C = (c_{ij})$ as

$$c_{ij} := \begin{cases} a_{ij} + b_{ij} & \text{if } \{i, j\} \cap R = \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

and $A = (a_{ij})$ respectively $B = (b_{ij})$ as

$$a_{ij} := \begin{cases} a_{ij} & \text{if } \{i, j\} \cap R \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \qquad b_{ij} := \begin{cases} b_{ij} & \text{if } \{i, j\} \cap R \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

Now let (σ, f, u, v) be a stable outcome for the instance A, B, C. Assume that for $\{i, \sigma(i)\} \cap R = \emptyset$ we have $f(i) = 0$. By the definition of an outcome we have $u_i = a_{i\sigma(i)} = 0 = v_{\sigma(i)} = b_{i\sigma(i)}$. By the stability of the solution this implies

$u_i + v_{\sigma(i)} = 0 \geq c_{i\sigma(i)} \geq 0$. Hence, $c_{i\sigma(i)} = 0$ and modifying f to $f(i) = 1$ maintains stability. If on the other hand for $\{i, \sigma(i)\} \cap R \neq \emptyset$ we have $f(i) = 1$ then $u_i + v_{\sigma(i)} = c_{i\sigma(i)} = 0$ and thus $u_i = v_{\sigma(i)} = 0$. By the stability of the solution, this implies $a_{i\sigma(i)} = 0$ or $b_{i\sigma(i)} = 0$. If only one of $a_{i\sigma(i)}$ and $b_{i\sigma(i)}$ is zero, say $a_{i\sigma(i)} > 0$, we may change u_i to $a_{i\sigma(i)}$ and set $f(i) = 0$ without changing the stability of the outcome. In addition the total payoff of the game increases.

Having dealt with these degenerate situations, we may assume that $f(i) = 1$ if and only if $\{i, \sigma(i)\} \cap R = \emptyset$ and hence, our outcome is also an outcome for the Eriksson-Karlander model. Clearly, stability in our model implies stability in the Eriksson-Karlander model.

3 An Algorithm to Find a Stable Outcome

The basic idea of our algorithm is derived from the (pseudopolynomial) auction procedure of Eriksson and Karlander. Hochstättler, Nickel and Jin [5] turned this into an $\mathcal{O}(n^4)$ algorithm. Schiess [10] generalized that algorithm to his "Decisive Edges Model" which is another special case of our model.

It will make the description of the algorithm easier, if we, at least partially, describe it in terms of an economic interpretation. We will call the elements from P *firms* and the players from Q *workers*. A rigid edge between a firm and a worker may be interpreted as an employment that is payed according to tariffs negotiated by some unions and flexible edges are payed according to individual contracts.

During the algorithm we maintain a (partial) map $\tau : P \to Q$ which we will turn into a permutation. If $\tau(i) = j$ we say that *firm i proposes to worker j* and that *j has a proposal*. For each firm $i \in P$ we have a set of possible proposals during the different stages of the algorithm. This defines a bipartite graph of *feasible proposals* which depends on the current map $v : Q \to \mathbb{R}_+$ of (expected) income of the workers. Additionally, we maintain a flexibility function $f : P \to \{0, 1\}$ that records whether a proposal refers to the flexible or the rigid edge.

The algorithm is a generalization of the classic "men propose – women dispose" algorithm for the stable marriage problem [3]. In the first stage – PlaceProposals in Algorithm 1, see also Algorithm 2 – each firm proposes to a worker who maximizes its expected profit, if it is non-zero, i.e. the firm is *solvent*. Otherwise, the firm is *insolvent*, i.e. in all stable solutions its payoff will be zero. We neglect it until the very end of the algorithm where we map it to an arbitrary worker without proposal. Workers with a rigid proposal, that additionally is the best offer they have got, dispose all other rigid proposals. Firms whose (rigid) proposal has been disposed propose to the next best worker. This is iterated until each worker has at most one rigid proposal. Then, using augmenting path techniques, we try to increase the set of workers that have a proposal (line 5 - 10 of Algorithm 1). If the map of the proposals still is not injective, we increase the (expected) income of workers with a flexible proposal. This is done in line

11 of Algorithm 1, see also Algorithm 3, similarly to the dual update step in the Hungarian Method for the assignment game [7]. Further proposals become feasible and we proceed with the first stage omitting the proposal placements for firms with a proposal.

To be more precise: initially, we set $v = 0$. In line 4 of Algorithm 2 for each firm $i \in P$ we determine a worker $j \in Q$ who maximizes the firm's expected income, i.e. such that

$$\max_{k=1}^{n}\{c_{ik}, a_{ik}\} \in \{c_{ij}, a_{ij}\},$$

and set $f(i) = 0$ if $\max_{k=1}^{n}\{c_{ik}, a_{ik}\} = a_{ij}$ and $f(i) = 1$ otherwise. If there are ties we prefer a_{ik}s to $c_{ik'}$s, further ties are broken arbitrarily. We make a proposal only if the profit is strictly positive. Firms which cannot yield any positive profit remain without proposal until the very end of the algorithm. If a worker $j \in P$ has a rigid proposal we set

$$v_j = \max\{b_{i\tau(i)} \mid \tau(i) = j, f(i) = 0\}$$

and possibly dispose all further rigid proposals to j by undefining $\tau(k)$ and $f(k)$.

Given v, τ and f we consider the map $u_{\tau,v} : \{0,1\} \times P \times Q \to \mathbb{R}_+$ defined by

$$u_{\tau,v}(g,i,j) = \begin{cases} c_{ij} - v_j & \text{if } g = 1 \\ a_{ij} & \text{if } g = 0 \text{ and } (v_j < b_{ij} \text{ or } \tau(i) = j) \\ 0 & \text{otherwise }. \end{cases}$$

Note that for fixed v this is the profit that firm i can expect when worker j is hired. If $v_j \geq b_{ij}$ and $\tau(i) \neq j$ then j will not accept a rigid proposal that does not yield a strictly larger income.

For each firm $i \in P$ we select j and $g \in \{0,1\}$ such that

$$u_{\tau,v}(g,i,j) = \max_{k=1}^{n}\max_{s=0}^{1} u_{\tau,v}(s,i,k). \tag{3}$$

Again, if possible we choose j such that $u_{\tau,v}(0,i,j) = \max_{k=1}^{n}\max_{s=0}^{1} u_{\tau,v}(s,i,k)$ and set $\tau(i) = j$. This makes τ a map again (up to insolvent firms). We iterate this process, taking additionally into account that a worker j with $v_j = b_{i^*j}$ where i^* is the favorite rigidly proposing firm of j and with at least one flexible proposal will dispose all rigid proposals, until each worker either has one or no rigid proposal.

Then we proceed to the next stage and consider the following (bipartite) digraph on $P\dot\cup Q$. We have the *backward edges* $(\tau(i), i)$ for all $i \in P$ and forward edges

$$\begin{aligned} D^i_{\tau,v} \quad := \quad & \left\{(i,j) \mid a_{ij} = u_{\tau,v}(0,i,j) = \max_{k=1}^{n}\max_{s=0}^{1} u_{\tau,v}(s,i,k) > 0, \tau(i) \neq j\right\} \\ \cup \quad & \left\{(i,j) \mid c_{ij} - v_j = u_{\tau,v}(1,i,j) = \max_{k=1}^{n}\max_{s=0}^{1} u_{\tau,v}(s,i,k) > 0, \tau(i) \neq j\right\}. \end{aligned}$$

Thus, the edges of the digraph correspond to rigid and flexible proposals (backward arcs) and to forward arcs that promise maximal profit, from the point of view of the firms. If this digraph contains a directed path from a worker with several proposals that ends in a rigid edge, an insolvent firm, a worker without proposal or to a worker with a rigid proposal, we invert this path and modify τ and f such that the corresponding combination of the maps is represented by the backward edges again. In the last case we additionally dispose the old rigid proposal (line 6 in Algorithm 1). We iterate this process until such a dipath does no longer exist. If there is still a worker with more than one proposal, we enter the third stage (lines 11 and 12 in Algorithm 1). Otherwise, we arbitrarily map unmapped firms to unmapped workers, set $u_i = \max_{k=1}^{n} \max_{s=0}^{1} u_{\tau,v}(s,i,k)$ and terminate.

Algorithm 1 The Main Loop

1: $v \leftarrow 0$
2: PlaceProposals
3: Construct digraph of feasible proposals
4: **while** there exists $j_0 \in Q$ with more than one proposal **do**
5:     **while** there exists a directed path $\mathcal{P}$ with at least two edges from j_0 to $j_1 \in Q$ where $\mathcal{P}$ ends in its only rigid edge or ends in an insolvent firm or j_1 has no proposal or j_1 has a rigid proposal **do**
6:         DisposeRigid(j_1)
7:         Alternate($\mathcal{P}$)
8:         PlaceProposals
9:         Update digraph, τ and f
10:     **end while**
11:     HungarianUpdate
12:     Update digraph, τ and f
13: **end while**
14: **while** there exists a firm $i \in P$ without proposal **do**
15:     Choose $j \in Q$ without proposal
16:     $\tau(i) \leftarrow j$
17:     $v_j \leftarrow b_{ij}$
18:     $f(i) \leftarrow 0$
19: **end while**
20: **for all** $i \in P$ **do**
21:     $u_i \leftarrow u_{\tau,v}(f(i), i, \tau(i))$
22: **end for**

Now, given a worker with more than one proposal, assuming that a dipath as required in line 5 in Algorithm 1 does no longer exist, consider its component in the graph underlying the digraph of feasible proposals. Denote by $\bar{P}$ the firms and by $\bar{Q}$ the workers in this component. Now compute

$$\begin{aligned}
u_i &:= \max_{k=1}^{n} \max_{s=0}^{1} u_{\tau,v}(s,i,k) \quad \forall i \in \bar{P} \\
\Delta_1 &:= \min \left\{ u_i - \max_{s=0}^{1} u_{\tau,v}(s,i,j) \mid i \in \bar{P}, j \notin \bar{Q} \right\} \\
\Delta_2 &:= \min \left\{ u_i - u_{\tau,v}(0,i,j) \mid i \in \bar{P}, j \in \bar{Q} \right\} \\
\Delta_3 &:= \min \left\{ u_i \mid i \in \bar{P} \right\}
\end{aligned}$$

and

$$\Delta := \min\{\Delta_1, \Delta_2, \Delta_3\} > 0 \text{ and set } v_j = v_j + \Delta \text{ for all } j \in \bar{Q}.$$

This way at least one new forward arc enters the digraph of proposals (backward arcs) and feasible proposals (forward arcs) or a firm is marked insolvent. We update the digraph and proceed with stage one.

Algorithm 2 PlaceProposals

```
procedure PlaceProposals
    while there exists a solvent firm i ∈ P without proposal do
        while there exists a solvent firm i ∈ P without proposal do
            Propose(i)
        end while
        for all j ∈ Q with a rigid proposal do
            Let i* ∈ P be a favorite rigid proposer in τ^{-1}(j)
            v_j ← b_{i*j}
            Update flexible proposals
            if j has no flexible proposal then
                Dispose all other rigid proposals
            else
                Dispose all rigid proposals
            end if
        end for
    end while
end procedure
```

4 Correctness of the Algorithm

First, note that all statements are feasible. In particular in the routine Propose(i) we can always select $j \in Q$ as described in (3) if $j \in Q$ has any feasible proposal given current v at all.

For a proof of correctness we make the following observations.

Proposition 1. 1. *The routine* PlaceProposals *never decreases* $|\tau(P)|$.

2. *If* $|\tau(P)|$ *is decremented by the statement* DisposeRigid(j_1) *then it is immediately incremented again by* Alternate$(\mathcal{P})$.

Algorithm 3 HungarianUpdate

```
1: procedure HungarianUpdate
2:     Choose j ∈ Q with several proposals
3:     Determine the vertices P̄ ⊆ P and Q̄ ⊆ Q in the component of j
4:     for all i ∈ P̄ do
5:         u_i ← max_{k=1}^n max_{s=0}^1 u_{τ,v}(s, i, k)
6:     end for
7:     Δ_1 ← min {u_i − max_{s=0}^1 u_{τ,v}(s, i, z) | i ∈ P̄, z ∈ Q \ Q̄}
8:     Δ_2 ← min {u_i − u_{τ,v}(0, i, j) | i ∈ P̄, j ∈ Q̄}
9:     Δ_3 ← min {u_i | i ∈ P̄}
10:    Δ ← min{Δ_1, Δ_2, Δ_3}
11:    for all j ∈ Q̄ do
12:        v_j ← v_j + Δ
13:    end for
14: end procedure
```

3. *The function $v : Q \to \mathbb{R}_+$ never decreases during the algorithm. If $v_j > 0$ then j has a proposal.*

4. *A disposed rigid proposal will never be proposed again.*

Proof.

1. In PlaceProposals τ is changed only in lines 4, 11 and 13 of Algorithm 2. Clearly, $|\tau(P)|$ does decrease in neither of these.

2. We dispose a rigid edge to a worker j_1 if we have found a directed path from a worker with at least two proposals to j_1. Denote by (i_1, j_1) the last edge in this path. In Alternate($\mathcal{P}$) we invert this edge and thus reinstall the old set $\tau(P)$.

3. v_j is changed in line 8 of Algorithm 2 and line 12 of Algorithm 3. In both cases v_j does not decrease and j has a proposal. A firm changes its proposal only if either its old rigid proposal has been disposed or if a path is alternated. Thus, a worker j with $v_j > 0$ will always have a proposer. When we modify v_j in line 17 of Algorithm 1 j had no proposal and thus, v_j was zero before.

4. When a rigid proposal $\tau(i) = j$ is disposed in line 11 or 13 of Algorithm 2 we have $b_{ij} \leq v_j$ and since v is non-decreasing in the following we always have $u_{\tau,v}(0, i, j) = 0$, so this edge will never be proposed again. Now assume we dispose a rigid edge in line 6 of Algorithm 1. Let again (i_1, j_1) denote the last edge on $\mathcal{P}$. When i_1 was mapped to j_1 we set $v_{j_1} = b_{i_1 j_1}$ since v_{j_1} is non-decreasing and by definition of $u_{\tau,v}$ the rigid edge will never be proposed again.

The following properties of the digraph of feasible proposals will be useful.

Proposition 2. *Consider the situation when the Algorithm enters the* HungarianUpdate.

1. *If a worker has more than one proposal then he has no rigid proposal.*
2. *$D^i_{\tau,v}$ consists only of flexible edges for all $i \in \bar{P}$. Furthermore, $\tau(i)$ refers to a flexible edge for all $i \in \bar{P}$.*

Proof. The first assertion follows from line 10 – 14 of Algorithm 2. If $D^i_{\tau,v}$ contains a rigid edge or $\tau(i)$ refers to a rigid edge, then there can be no directed path from j_0 to i for otherwise we would not have left the inner while-loop and hence $i \notin \bar{P}$. □

Theorem 1. *Algorithm 1 terminates after a finite number of steps.*

Proof. In the inner while loop (line 5 – 10) we either make a new rigid proposal, dispose a rigid proposal in line 6, mark a new firm as insolvent or, if $\mathcal{P}$ ends in an unmapped worker, increase $|\tau(P)|$.

If there is no more such path $\mathcal{P}$, we call the routine HungarianUpdate. Note that in the beginning of that procedure no vertex in $\bar{P} \cup \bar{Q}$ is incident with a rigid edge by Proposition 2 and hence, $\Delta > 0$. Let (s, i_0, z_0) be such that $\Delta = \Delta_1 = u_{i_0} - u_{\tau,v}(s, i_0, z_0)$ with $i_0 \in \bar{P}, z_0 \in Q$ and $(i_0, z_0) \notin D^{i_0}_{\tau,v}$. We will show that with the dual update in line 11 – 13 of Algorithm 3 we do not lose an edge in the digraph of feasible proposals but add (i_0, z_0). Clearly, we do not lose any backward edge. Thus, assume $(i, j) \in D^i_{\tau,v}$ was a forward edge of the digraph before the dual update. Hence,

$$u_{\tau,v}(s, i, j) = \max_{k=1}^{n} \max_{s=0}^{1} u_{\tau,v}(s, i, k) > 0 \text{ and } \tau(i) \neq j. \tag{4}$$

This value is changed in the dual update since $u_{\tau,v}(1, i, j) = c_{ij} - v_j$ (recall that all "old" edges in the component where flexible by Proposition 2). But it is changed by the same amount for all edges connecting i to some $k \in \bar{Q}$ and by definition of Δ_1 we still have $u_{\tau,v}(s, i, j) = \max_{k=1}^{n} \max_{s=0}^{1} u_{\tau,v}(s, i, k)$ after the update. Hence, (i, j) is still an edge of the digraph after the update.

Finally, since

$$\Delta_1 = \max_{k=1}^{n} \max_{s=0}^{1} u_{\tau,v}(s, i_0, k) - \max_{s=0}^{1} u_{\tau,v}(s, i_0, z_0)$$

we must have $(i_0, z_0) \in D^{i_0}_{\tau,v}$ after the update. Since by Proposition 1 a disposed rigid edge never will be proposed again, we can add each rigid edge at most once and each firm is marked insolvent at most once. Also after adding sufficiently many edges a path as required must occur. When $\Delta = \Delta_2$ a new rigid edge enters the digraph. This can happen at most n^2 times. When $\Delta = \Delta_3$ a firm is marked insolvent. This can happen at most n times. Hence, the algorithm terminates after a finite number of steps. □

Theorem 2. *Algorithm 1 computes a stable outcome.*

Proof. When the algorithm terminates, no worker has two proposals and all firms have a proposal, thus τ is a bijection, or a *matching permutation*. v is a non-negative function and for all rigid proposals $\tau(i) = j$ we have set $v_j = b_{ij}$. Note that, when v is updated in line 12 of Algorithm 3 by Proposition 2, no worker in $\bar{Q}$ is incident with a rigid edge in the graph of feasible proposals. Before finishing the algorithm we set u_i to $a_{i\tau(i)}$ if $f(i) = 0$ and to $c_{i\tau(i)} - v_{\tau(i)}$ otherwise. Hence, the algorithm computes an outcome.

In order to show that the outcome is stable we consider the development of the function

$$\bar{u} : P \to \mathbb{R}_+ \text{ defined by } \bar{u}_i := \max_{k=1}^{n} \max_{s=0}^{1} u_{\tau,v}(s, i, k).$$

By definition the pair $(\bar{u}, v)$ satisfies the stability conditions (1) and (2). Also if $\tau(i)$ is defined we have $\bar{u}_i = u_{\tau,v}(f(i), i, \tau(i))$. Hence, when we terminate $\bar{u} = u$ holds. It follows that the algorithm produces a stable outcome. □

5 Running Time Analysis

Theorem 3. *The algorithm terminates in $\mathcal{O}(n^4)$.*

Proof. Lines 14 –19 in Algorithm 1 are easily implemented in $\mathcal{O}(n^2)$ and lines 20 – 22 in Algorithm 1 in linear time. PlaceProposals can be implemented with the same complexity as the classic "Men-Propose-Women-Dispose" algorithm and thus in $\mathcal{O}(n^2)$. Hence we may focus on the nested while-loops. In the inner while-loop we either increment $|\tau(P)|$ which is possible at most n times or dispose a rigid edge of which there are only n^2, introduce a new egde into the graph of feasible proposals also at most n^2 times or mark one of n firms insolvent. The complexity of a single inner while loop is dominated by the procedure PlaceProposals. Thus, the inner while loop in total accounts for $\mathcal{O}(n^4)$.

Finally, if HungarianUpdate introduces new flexible edges we may continue our search on the augmented data. A standard search procedure requires $\mathcal{O}(|E|) = \mathcal{O}(n^2)$ if E denotes the set of edges in our graph. This is multiplied at worst with the number of rigid edges which are newly introduced into the digraph of feasible proposals or are disposed, the number of augmentations of $\tau(P)$ and the number of firms that are marked insolvent. Hence, it accounts for $\mathcal{O}(n^4)$ in total as well.

Altogether this sums up to $\mathcal{O}(n^4)$. □

Remark 2. *Note that our algorithm in fact runs in quadratic time as the size of the input data is $\mathcal{O}(n^2)$.*

Acknowledgments. We thank Elisabeth Gassner for asking the right question.

References

[1] G. Demange, D. Gale, and M. Sotomayor, *Multi-item auctions*, Journal of Political Economy, 94 (1986), pp. 863–872.

[2] K. Eriksson and J. Karlander, *Stable matching in a common generalization of the marriage and assignment models*, Discrete Mathematics, 217 (2000), pp. 135–156.

[3] D. Gale and L. S. Shapley, *College admissions and the stability of marriage*, American Mathematical Monthly, 69 (1962), pp. 9–15.

[4] D. Gusfield and R. W. Irving, *The stable marriage problem: Structure and algorithms*, MIT Press, Cambridge, MA, USA, 1989.

[5] W. Hochstättler, H. Jin, and R. Nickel, *Note on an auction procedure for a matching game in polynomial time*, in AAIM, 2006, pp. 387–394.

[6] D. E. Knuth, *Stable marriage and its relation to other combinatorial problems*, in CRM Proceedings and Lecture Notes, vol. 10, American Mathematical Society, 1997.

[7] H. W. Kuhn, *The Hungarian method for the assignment problem*, Naval Research Logistics Quaterly, 2 (1955), pp. 83–97.

[8] A. E. Roth and M. Sotomayor, *Two-sided matching: A study in game-theoretic modeling and analysis*, Cambridge University Press, Cambridge, 1991.

[9] A. E. Roth and M. Sotomayor, *Stable outcomes in discrete and continuous models of two-sided matching: A unified treatment*, Revista de Econometria, The Brazilian Review of Econometrics, 16 (1996).

[10] D. Schiess, *Mixed Matching Markets.* Bachelor's Thesis, FernUniversität in Hagen, Germany, August 2007.

[11] L. S. Shapley and M. Shubik, *The assignment game I: The core*, International Journal of Game Theory, 1 (1972), pp. 111–130.