Discrete Applied Mathematics 157 (2009) 3181–3186

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Note

Optimal popular matchings[☆]

Telikepalli Kavitha [1], Meghana Nasre [*]

Indian Institute of Science, Bangalore, India

ARTICLE INFO

Article history:
Received 16 February 2009
Received in revised form 1 May 2009
Accepted 3 June 2009
Available online 26 June 2009

Keywords:
Design of algorithms
Bipartite graphs
Matchings
One-sided preference lists

ABSTRACT

In this paper we consider the problem of computing an "optimal" popular matching. We assume that our input instance $G = (\mathcal{A} \cup \mathcal{P}, E_1 \dot{\cup} \cdots \dot{\cup} E_r)$ admits a *popular matching* and here we are asked to return not any popular matching but an *optimal* popular matching, where the definition of optimality is given as a part of the problem statement; for instance, optimality could be *fairness* in which case we are required to return a *fair* popular matching. We show an $O(n^2 + m)$ algorithm for this problem, assuming that the preference lists are strict, where m is the number of edges in G and n is the number of applicants.

1. Introduction

In this paper we consider the problem of computing an *optimal* popular matching in a bipartite graph $G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E})$ with one-sided preference lists. Optimality is described succinctly as a part of the problem statement, for instance, *rank-maximality*, *fairness*, or *minimum cost of matched edges* can be considered as optimality. The algorithm that we present, in fact, works for several notions of optimality. We consider the problem of computing a popular matching M in G such that no popular matching is *more optimal* than M. We first describe below the popular matching problem.

1.1. The popular matching problem

An instance of the *popular matching problem* is a bipartite graph $G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E})$ and a partition $\mathcal{E} = E_1 \dot{\cup} E_2 \cdots \dot{\cup} E_r$ of the edge set. The vertices of $\mathcal{A}$ are called *applicants* and the vertices of $\mathcal{P}$ are called *posts*. For each $1 \leq i \leq r$, the elements of E_i are called the edges of rank i. If $(a, p) \in E_i$ and $(a, p') \in E_j$ with $i < j$, we say that a prefers p to p'. This ordering of posts adjacent to a is called a's preference list. For any applicant a and any rank i, where $1 \leq i \leq r$, we assume that there is at most one post p such that $(a, p) \in E_i$, that is, we assume that preference lists are strictly ordered.

A *matching* M of G is a set of edges such that no two edges share an endpoint. We denote by $M(a)$ the post to which applicant a is matched in M. We say that an applicant a prefers matching M' to M if (i) a is matched in M' and unmatched in M, or (ii) a is matched in both M' and M, and a prefers $M'(a)$ to $M(a)$. M' is *more popular than* M if the number of applicants that prefer M' to M exceeds the number of applicants that prefer M to M'. A matching M^* is *popular* if there is no matching that is more popular than M^*.

[☆] A preliminary version of this work appeared in the workshop MATCH-UP: Matching Under Preferences (http://www.optimalmatching.com/workshop), July 2008, Reykjavik, Iceland.

[*] Corresponding address: Computer Science and Automation Department, Indian Institute of Science, Bangalore 560012, India. Tel.: +91 80 22932368; fax: +91 80 23602911.

E-mail addresses: kavitha@csa.iisc.ernet.in (T. Kavitha), meghana@csa.iisc.ernet.in (M. Nasre).

[1] Work done as a part of the DST-MPG partner group on Efficient Graph Algorithms, IISc Bangalore.

0166-218X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2009.06.004

The *popular matching problem* is to determine if a given instance admits a popular matching, and to find such a matching, if one exists. The popular matching problem was considered by Abraham et al. in [1] and a linear time algorithm (for the case of strictly ordered preference lists) was given to determine if G admits a popular matching and to compute a maximum-cardinality popular matching.

1.2. Problem definition

In this paper we assume that the input instance G admits a popular matching and here we are not content in returning any popular matching or any maximum-cardinality popular matching. Our goal is to compute an *optimal* popular matching, where the definition of optimality is given succinctly as a part of the problem definition. For instance, the problem description could state *fairness* as optimality, which means that, among all popular matchings in G, we have to return that popular matching which is the most *fair*.

The fair matching problem in a bipartite graph $G = (\mathcal{A} \cup \mathcal{P}, E_1 \dot{\cup} E_2 \cdots \dot{\cup} E_r)$ asks for a matching M that satisfies the properties below: (i) M is a maximum-cardinality matching in G, and (ii) among all maximum-cardinality matchings in G, M matches the least number of applicants to their rank r posts, subject to this constraint, matches the least number of applicants to their rank $r-1$ posts, subject to this constraint, matches the least number of applicants to their rank $r-2$ posts, and so on. Currently, there are no purely combinatorial algorithms known for computing a fair matching and the best algorithm for the fair matching problem reduces this to the minimum weight maximum-cardinality matching problem by assigning weight n^{k-1} to a rank k edge for each $1 \leq k \leq r$.

For convenience, as was done in [1], we will add a dummy post ℓ_a at the end of a's preference list, for each applicant a, and assign the edge (a, ℓ_a) rank $r+1$. Thus henceforth, the edge set $\mathcal{E} = E_1 \dot{\cup} \cdots \dot{\cup} E_{r+1}$ and any unmatched applicant a will be assumed to be matched to ℓ_a. So all matchings are always applicant-complete from now. A fair popular matching can now be defined as follows.

Definition 1. A popular matching M in G that matches the least number of applicants to their rank $r+1$ posts, subject to this constraint, matches the least number of applicants to their rank r posts, subject to this constraint, matches the least number of applicants to their rank $r-1$ posts, and so on, is a *fair popular* matching.

Other notions of optimality include *rank-maximality*[2] [4], or *min-cost* of matched edges (each edge here has a cost associated with it). Analogous to a fair popular matching, we can define a *rank-maximal popular* matching or a *min-cost popular* matching.

Our optimality criteria: Recall that we assumed the optimality criterion O is specified as a part of the problem. For any two matchings M_1, M_2, let $M_1 \leq_O M_2$ stand for either $M_1 <_O M_2$ (that is, M_1 is less optimal than M_2) or $M_1 \approx_O M_2$ (that is, M_1 and M_2 are as optimal as each other). We need the following properties.

(A) $\leq_O$ is complete, that is, for any pair of matchings M_1, M_2 either $M_1 \leq_O M_2$ or $M_2 \leq_O M_1$. In fact, exactly one of (i), (ii), (iii) holds: (i) $M_1 <_O M_2$, (ii) $M_2 <_O M_1$, (iii) $M_1 \approx_O M_2$.
(B) $\leq_O$ is transitive: that is, $M_1 \leq_O M_2$ and $M_2 \leq_O M_3 \Rightarrow M_1 \leq_O M_3$.
(C) If an edge e belongs to two matchings M_i, M_j, then $M_i <_O M_j \Leftrightarrow M_i - \{e\} <_O M_j - \{e\}$. Note that this implies that if e belongs to M_i and M_j, then $M_i \approx_O M_j \Leftrightarrow M_i - \{e\} \approx_O M_j - \{e\}$.

Note that optimality criteria like rank-maximality, fairness, min-cost of matched edges satisfy these properties. In fact, any natural optimality criterion that one defines in practice would satisfy the above properties.

1.3. Related results

The notion of popular matchings was originally introduced by Gardenfors [2] in the context of the stable marriage problem with two-sided preference lists. In the case of two-sided preference lists where the two sides of the bipartite graph are considered *men* and *women*, a stable matching is considered the ideal answer to what is a desirable matching. However there is a wide spectrum of stable matchings ranging from men-optimal stable matchings to women-optimal stable matchings. Irving, Leather, and Gusfield [5] considered the problem of computing a stable matching that is optimal under some more equitable criterion of optimality. In fact, much work has been done in the two-sided preference list setting on finding stable matchings that satisfy additional criteria (see [3] for an overview). In the same vein, assuming that the input instance G admits a popular matching, here we ask for an *optimal* popular matching where optimality is defined as a part of the problem statement.

The problem of computing a fair popular matching/rank-maximal popular matching/min-cost popular matching can be solved by assigning suitable costs to the edges of an appropriate bipartite graph H derived from $G = (\mathcal{A} \cup \mathcal{P}, E_1 \dot{\cup} \cdots \dot{\cup} E_r)$ and computing a min-cost or max-cost perfect matching in H. Here we present a simple combinatorial algorithm that runs

[2] A matching M is rank-maximal if M matches the maximum number of applicants to their rank 1 posts, subject to this constraint M matches the maximum number of applicants to their rank 2 posts, and so on.

in $O(n^2 + m)$ time for the problem of computing an "optimal" popular matching in a bipartite graph where m is the number of edges and n is the number of applicants. We assume that given two matchings M_1 and M_2, we can determine if $M_1 <_O M_2$, $M_2 <_O M_1$ or $M_1 \approx_O M_2$ in $O(n)$ time; this is a reasonable assumption which is indeed true for fairness, rank-maximality, or min-cost.

Very recently, McDermid and Irving [7] also considered the optimal popular matchings problem in $G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E})$ with strict preference lists. They showed an $O(m + n \log n)$ algorithm for computing fair popular matchings and rank-maximal popular matchings and an $O(m + n)$ algorithm for min-cost popular matchings. Their main tool was a graph called the *switching graph*, used by Mahdian [6] to investigate the probability of the existence of popular matchings in random graphs. Though our algorithm is slower, our algorithm is extremely simple. Our algorithm is iterative: in the ith iteration, it adds the ith applicant a_i to the current graph on applicants $a_1, \ldots, a_{i-1}$ and augments the current matching on $a_1, \ldots, a_{i-1}$ to a *most optimal* matching that matches all of $a_1, \ldots, a_i$ and all their top choice posts. We show that this yields the desired matching at the end of n iterations. It is indeed interesting that a method as simple as this works. The problem of computing an optimal popular matching is not only of theoretical interest but also of practical importance and our algorithm is useful for such applications because it can be implemented very easily.

2. Preliminaries

In this section we review the algorithmic characterisation for computing a popular matching from [1]. Since our problem is restricted to the case where preference lists do not have ties, we will present the characterisation from [1] of popular matchings for strictly ordered preference lists.

For each applicant a, define a *first choice post* for a, denoted by $f(a)$, and a *second choice post* for a, denoted by $s(a)$, as follows. The post $f(a)$, is one that occurs at the top of a's preference list, that is, it is a's most preferred post. The post $s(a)$ is the most preferred post on a's list that is *not* $f(a')$ for any applicant a'. Note that by the above definition, f-posts are disjoint from s-posts. For each applicant a, $f(a)$ is guaranteed to exist if its preference list is non-empty. Note that the dummy post ℓ_a added at the end of a's preference list ensures that $s(a)$ always exists for each applicant a.

The following lemma from [1] characterises a popular matching.

Lemma 1. *A matching M is popular if and only if*

(1) *every f-post is matched in M,*
(2) *for each applicant a, $M(a) \in \{f(a), s(a)\}$.*

Let $G'(A \cup P, E')$ denote the graph in which each applicant a has exactly two edges, $(a, f(a))$ and $(a, s(a))$ incident to it. From Lemma 1, it is immediate that the input instance G admits a popular matching if and only if the graph G' defined above admits an $\mathcal{A}$-perfect matching. Using this characterisation, a linear time algorithm was presented in [1] to determine if G admits a popular matching and compute one, if it exists.

3. Our algorithm

In this section we describe our algorithm to compute an *optimal* popular matching in G with respect to the optimality criterion specified as a part of the input. Any optimal popular matching, by the virtue of being popular, needs to satisfy the properties specified in Lemma 1. So we can operate on a reduced graph $G = (\mathcal{A} \cup \mathcal{P}, E')$ where E' consists of edges $(a, f(a))$ and $(a, s(a))$ for each $a \in \mathcal{A}$. Hence from now on we can assume that every applicant in G has degree at most 2.

Let n be the number of applicants in G and $a_1, \ldots, a_n$ be an arbitrary ordering of the applicants. We denote by H_k, for $1 \le k \le n$, the graph on vertex set $\{a_1, \ldots, a_k\} \cup \{f(a_1), \ldots, f(a_k), s(a_1), \ldots, s(a_k)\}$ and edges $\{(a_j, f(a_j)), (a_j, s(a_j))\}$, for $1 \le j \le k$. We present a simple iterative strategy for computing an optimal popular matching in G. For each $1 \le k \le n$, we will compute a matching M_k that satisfies the following 2 properties: (1) M_k is a matching of size k in H_k that matches all the posts $f(a_1), \ldots, f(a_k)$; (2) among all matchings that satisfy (1), M_k is an optimal matching.

We compute matching M_k satisfying the above mentioned properties iteratively. Say we have already computed the desired matching M_{k-1} in the graph H_{k-1}. We add to the graph H_{k-1} the applicant a_k, the posts $f(a_k)$, $s(a_k)$ if they do not exist and the edges $(a_k, f(a_k))$ and $(a_k, s(a_k))$ to form the graph H_k. We will show that M_k can be computed by *augmenting* M_{k-1} appropriately. Since the matching M_k has to be popular, M_k has to match each of the applicants $a_1, \ldots, a_k$: due to the fact that we augment M_{k-1} in H_k, each of $a_1, \ldots, a_{k-1}$ remains matched (to either its f-post or s-post). Also since M_k needs to match a_k, either $(a_k, f(a_k))$ or $(a_k, s(a_k))$ has to belong to M_k. Our algorithm tries both the options:

(1) it tries to find augmenting paths p_k and q_k with respect to M_{k-1} in H_k in order to match a_k to $f(a_k)$ and to $s(a_k)$, respectively. We will show that at least one of p_k, q_k has to exist.

(2) If p_k does not exist, then $M_k = M_{k-1} \oplus q_k$ and if q_k does not exist, then $M_k = M_{k-1} \oplus p_k$. If both p_k and q_k exist, then the more optimal of $M_{k-1} \oplus p_k$ and $M_{k-1} \oplus q_k$ is chosen as M_k. Theorem 1 shows that this simple method suffices.

Our algorithm is presented as Algorithm 3.1.

Theorem 1. *The matching M_n returned by our algorithm is a popular matching that is maximal w.r.t. O.*

Algorithm 3.1 Our algorithm to compute an optimal popular matching

– Set any order among the applicants so that the applicants can be labelled $a_1, a_2, \ldots, a_n$.
– Let H_1 be the graph on vertex set $\{a_1\} \cup \{f(a_1), s(a_1)\}$ and edge set $\{(a_1, f(a_1)), (a_1, s(a_1))\}$; let M_1 be the matching $\{(a_1, f(a_1)\}$.
– Initialize $i = 2$.
while $i \leq n$ **do**
  Update H_{i-1} to H_i by adding the applicant a_i and posts $f(a_i)$, $s(a_i)$ (if they do not already exist) to the vertex set and the edges $(a_i, f(a_i))$ and $(a_i, s(a_i))$ to the edge set.
  if $f(a_i)$ is newly added **then**
    $M_i = M_{i-1} \cup \{(a_i, f(a_i))\}$.
  else
    find an augmenting path p_i with respect to M_{i-1} in H_i that begins with the edge $(a_i, f(a_i))$
    find an augmenting path q_i with respect to M_{i-1} in H_i that begins with the edge $(a_i, s(a_i))$
    if p_i (similarly, q_i) does not exist **then**
      $M_i = M_{i-1} \oplus q_i$ (resp., $M_{i-1} \oplus p_i$).
    else if both p_i and q_i exist **then**
      $M_i =$ the more optimal of $M_{i-1} \oplus p_i$ and $M_{i-1} \oplus q_i$. {*In case $M_{i-1} \oplus p_i \approx_O M_{i-1} \oplus q_i$, then M_i can be either.*}
    end if
  end if
  $i = i + 1$.
end while
– Return M_n.

Note that it is easy to see that the matching M_n returned by our algorithm is popular. First, for each i, M_i is a maximum-cardinality matching in H_i. Thus M_n is a maximum-cardinality matching in H_n; we know that $H_n = (\mathcal{A} \cup \mathcal{P}, E')$ admits an $\mathcal{A}$-perfect matching since the input instance admits a popular matching. Thus M_n is an $\mathcal{A}$-perfect matching. Also, by construction, we never let an f-post remain unmatched. Thus, M_n is an $\mathcal{A}$-perfect matching in G that matches all f-posts. Thus M_n is a popular matching in the input instance.

We now need to show that among all popular matchings, M_n is an optimal matching. We will prove this by induction: we will show that for each i, M_i is a matching of size i in H_i that matches all posts $f(a_1), \ldots, f(a_i)$ and amongst all such matchings, M_i is a most optimal one. Then it is immediate that M_n is an optimal popular matching.

Note that we can compare 2 matchings M, M' of H_i with respect to the optimality criterion O by extending each of M, M' to $\{a_1, \ldots, a_n\}$ by matching $\{a_{i+1}, \ldots, a_n\}$ to their last resort posts (of rank $r + 1$). Thus we can use the relative order w.r.t. O on matchings in the input instance to compare two matchings in H_i.

We will now show that for all $1 \leq i \leq n$, M_i is optimal in H_i subject to the constraint that M_i has to match all of $a_1, \ldots, a_i$ and $f(a_1), \ldots, f(a_i)$. The base case $i = 1$ is trivial. By induction hypothesis, we assume that M_{k-1} is optimal in H_{k-1} subject to the constraint that it has to match all of $a_1, \ldots, a_{k-1}$ and $f(a_1), \ldots, f(a_{k-1})$. Using this hypothesis, we will show that M_k is optimal in H_k subject to the constraint that it has to match all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$.

We consider two cases: (i) $f(a_k)$ is not present in H_{k-1} and (ii) $f(a_k)$ is present in H_{k-1}. We will now consider the first case, that is, $f(a_k)$ is not present in H_{k-1}, and show the following lemma.

Lemma 2. *$M_k = M_{k-1} \cup \{(a_k, f(a_k))\}$ is an optimal matching in H_k subject to the constraint that all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$ have to be matched.*

Proof. It is immediate from the definitions of M_{k-1} and M_k that M_k matches all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$. What remains to prove is that M_k is an optimal matching.

Suppose not, let N_k be such a matching in H_k that is more optimal than M_k. We know that $f(a_k)$ is not an f-post for any applicant in $\{a_1, \ldots, a_{k-1}\}$ (by virtue of the fact that $f(a_k)$ is not present in H_{k-1}). Since N_k has to satisfy the constraint that all f-posts in H_k are matched, it follows that $N_k(a_k) = f(a_k)$. Thus N_k and M_k agree on the edge $e = (a_k, f(a_k))$.

Since $M_k <_O N_k$ and both these matchings contain the edge e, it follows from our condition (c) on O that $M_k - \{e\} <_O N_k - \{e\}$. The matching $N_k - \{e\}$ matches all of $a_1, \ldots, a_{k-1}$ and $f(a_1), \ldots, f(a_{k-1})$ since N_k matches all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$. However we know that $M_k - \{e\}$, which is the same as M_{k-1} (recall that $M_k = M_{k-1} \cup \{e\}$) is an optimal matching in H_{k-1}, contradicting that $M_k - \{e\} <_O N_k - \{e\}$. This completes the proof of Lemma 2. □

We now deal with the case when $f(a_k)$ is present in H_{k-1}. In this case, we try to find augmenting paths p_k and q_k in H_k. Note that at least one of p_k, q_k has to exist since H_k admits a matching of size k (any $\mathcal{A}$-perfect matching of $(\mathcal{A} \cup \mathcal{P}, E')$ restricted to $a_1, \ldots, a_k$ is a matching of size k in H_k)—thus there has to exist an augmenting path with respect to the $(k - 1)$-sized matching M_{k-1} in H_k. Say, p_k exists and q_k does not exist. Then we show the following.

Lemma 3. *$M_{k-1} \oplus p_k$ is an optimal matching in H_k subject to the constraint that it has to match all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$.*

Proof. It is easy to see that $M_k = M_{k-1} \oplus p_k$ matches all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$. We need to show that M_k is an optimal matching. Suppose not and let N_k be such a matching that is more optimal than M_k.

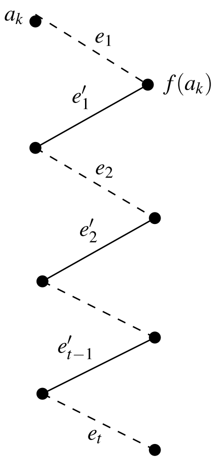

Fig. 1. The path p_k: the bold edges are present in M_{k-1} and the dashed edges are in M_k and in N_k.

We first claim that since q_k does not exist, any matching that matches all of $a_1, \ldots, a_k$ in H_k has to match a_k to $f(a_k)$. Suppose not and let L be a matching in H_k that matches all of $a_1, \ldots, a_{k-1}$ and also matches a_k to $s(a_k)$. Then consider that component of $L \oplus M_{k-1}$ which contains a_k. Since a_k is unmatched by M_{k-1}, this component is a path (call it q) that begins with the edge $(a_k, s(a_k))$. The existence of q contradicts our assumption that there was no augmenting path with respect to M_{k-1} in H_k that begins with the edge $(a_k, s(a_k))$.

The above claim implies that the matching N_k has to match a_k to $f(a_k)$. Thus both M_k and N_k agree on the edge $(a_k, f(a_k))$. Let a' be the applicant that was matched by M_{k-1} to $f(a_k) = f(a')$. Since $f(a_k)$ is matched to a_k by N_k, the applicant a' has to be matched to its s-post by N_k. Thus M_k and N_k also agree on the edge $(a', s(a'))$. In fact, *every* edge in p_k that is present in M_k has to be present in N_k. Thus N_k and M_k contain the same subset of edges of p_k. Call these edges $e_1, \ldots, e_t$ (refer to Fig. 1).

So if $M_k <_O N_k$, then $M_k - \{e_1, \ldots, e_t\} <_O N_k - \{e_1, \ldots, e_t\}$ since $e_1, \ldots, e_t$ are present in both N_k and M_k (refer to Fig. 1). By adding the edges $e'_1, e'_2, \ldots, e'_{t-1}$ (see Fig. 1) of $p_k - M_k$ to both $N_k - \{e_1, \ldots, e_t\}$ and $M_k - \{e_1, \ldots, e_t\}$, we get the matchings $N_k \oplus p_k$ and $M_k \oplus p_k$, since $e'_1, e'_2, \ldots, e'_{t-1}$ are also the edges of $p_k - N_k$. It follows that $N_k \oplus p_k$ is more optimal than $M_k \oplus p_k$ (by condition (c) on O). Now $N_k \oplus p_k$ is a matching in H_{k-1} that matches all of $a_1, \ldots, a_{k-1}$ and $f(a_1), \ldots, f(a_{k-1})$. However $M_k \oplus p_k = M_{k-1}$ is a most optimal such matching in H_{k-1}. Thus $N_k \oplus p_k$ cannot be more optimal than M_{k-1}, a contradiction. □

The case when q_k exists and p_k does not exist is absolutely similar to the above lemma. The only case that we are left with is the case when both p_k and q_k exist. In this case our algorithm computes both $M_{k-1} \oplus p_k$ and $M_{k-1} \oplus q_k$ and chooses the more optimal of these two matchings to be M_k. We now have to show that M_k is what we desire.

Lemma 4. *M_k, the more optimal matching between $M_{k-1} \oplus p_k$ and $M_{k-1} \oplus q_k$, is an optimal matching in H_k that matches all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$.*

Proof. It is obvious that M_k matches all of $a_1, \ldots, a_k$ and $f(a_1), \ldots, f(a_k)$. Suppose M_k is not an optimal such matching, let N_k be such a matching that is more optimal than M_k. The matching N_k has to match a_k to either $f(a_k)$ or to $s(a_k)$. We will show the following:

Claim 1. If $N_k(a_k) = f(a_k)$, then $N_k \leq_O M_{k-1} \oplus p_k$.
Claim 2. If $N_k(a_k) = s(a_k)$, then $N_k \leq_O M_{k-1} \oplus q_k$.

We know that either $N_k(a_k) = f(a_k)$ or $N_k(a_k) = s(a_k)$, which implies by Claims 1 and 2 that either $N_k \leq_O M_{k-1} \oplus p_k$ or $N_k \leq_O M_{k-1} \oplus q_k$. Because M_k is the more optimal of $M_{k-1} \oplus p_k$ and $M_{k-1} \oplus q_k$, we have: $M_{k-1} \oplus p_k \leq_O M_k$ and $M_{k-1} \oplus q_k \leq_O M_k$. It thus follows from the transitivity of $\leq_O$ that $N_k \leq_O M_k$. This contradicts our assumption that N_k is more optimal than M_k. Hence, what we need to show are Claims 1 and 2.

Proof of Claim 1

If $N_k(a_k) = f(a_k)$, then as we argued in the proof of Lemma 3, it follows that $M_{k-1} \oplus p_k$ and N_k contain the same subset of edges of p_k. Now consider $N_k \oplus p_k$: this is a matching in H_{k-1} that matches all of $a_1, \ldots, a_{k-1}$ and $f(a_1), \ldots, f(a_{k-1})$. Since M_{k-1} is an optimal matching in H_{k-1} that matches all of $a_1, \ldots, a_{k-1}$ and $f(a_1), \ldots, f(a_{k-1})$, it follows that $N_k \oplus p_k \leq_O M_{k-1}$. Hence by condition (c), $N_k = (N_k \oplus p_k) \oplus p_k \leq_O M_{k-1} \oplus p_k$. This finishes the proof of Claim 1.

The proof of Claim 2 is absolutely similar to the proof of Claim 1. □

This completes the proof of Theorem 1. We will now analyse the running time of Algorithm 3.1. The f- and s-posts of all applicants can be computed in $O(m)$ time. The main while loop of Algorithm 3.1 runs for n iterations and each iteration takes $O(n)$ time to construct the augmenting paths p_i, q_i and to compare $M_{i-1} \oplus p_i$ and $M_{i-1} \oplus q_i$. Thus our algorithm runs in $O(n^2 + m)$ time.

Acknowledgments

We thank the reviewers for their helpful comments.

References

[1] D.J. Abraham, R.W. Irving, T. Kavitha, K. Mehlhorn, Popular matchings, SIAM Journal on Computing 37 (4) (2007) 1030–1045.
[2] P. Gardenfors, Match making: Assignments based on bilateral preferences, Behavioural Sciences 20 (1975) 166–173.
[3] D. Gusfield, R.W. Irving, The Stable Marriage Problem: Structure and Algorithms, MIT Press, 1989.
[4] R.W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, K. Paluch, Rank-maximal matchings, ACM Transactions on Algorithms 2 (4) (2006) 602–610.
[5] R.W. Irving, P. Leather, D. Gusfield, An efficient algorithm for the optimal stable marriage, Journal of the ACM 34 (3) (1987) 532–543.
[6] M. Mahdian, Random popular matchings, in: Proceedings of the 7th ACM Conference on Electronic-Commerce, 2006, pp. 238–242.
[7] E. McDermid, R.W. Irving, Popular matchings: Structure and algorithms, in: University of Glasgow, Computing Science Department Research Report, TR-2008-292, 2008. To appear in COCOON 2009.