Rank-Maximal Matchings

Robert W. Irving[*] Telikepalli Kavitha[†] Kurt Mehlhorn[†]
Dimitrios Michail[‡] Katarzyna Paluch[§]

Abstract

Suppose that each member of a set $\mathscr{A}$ of applicants ranks a subset of a set $\mathscr{P}$ of posts in an order of preference, possibly involving ties. A *matching* is a set of (applicant, post) pairs such that each applicant and each post appears in at most one pair. A *rank-maximal* matching is one in which the maximum possible number of applicants are matched to their first choice post, and subject to that condition, the maximum possible number are matched to their second choice post, and so on. This is a relevant concept in any practical matching situation and it was first studied by Irving [8].

We give an algorithm to compute a rank-maximal matching with running time $O(\min(n+C, C\sqrt{n})m)$, where C is the maximal rank of an edge used in a rank-maximal matching, n is the number of applicants and posts and m is the total size of the preference lists.

1 Introduction

Let $\mathscr{A}$ be a set of *applicants* and $\mathscr{P}$ be a set of *posts*, and suppose that, associated with each member of $\mathscr{A}$ is a preference list (possibly involving ties) comprising a subset of the elements of $\mathscr{P}$. A *matching* of $\mathscr{A}$ to $\mathscr{P}$ is an allocation of each applicant to at most one post from his preference list so that each post is filled by at most one applicant; in other words it is a matching in the bipartite graph $G = (\mathscr{A} \cup \mathscr{P}, \mathscr{E})$, where $\mathscr{E}$ consists of all pairs (a, p) such that post p appears in the preference list of applicant a.

Each edge (a, p) has a rank i, which means that post p is an ith choice for applicant a. In any applicant a's list, there may be any number of ith choice posts, even zero. We believe that this is the natural way of formulating the problem of

[*]Department of Computing Science, University of Glasgow, UK. rwi@dcs.gla.ac.uk

[†]Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India, kavitha@csa.iisc.ernet.in. Work done while the author was at MPII.

[‡]Max-Planck-Institut für Informatik, Saarbrücken, Germany. {mehlhorn, michail}@mpi-sb.mpg.de. Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

[§]Institute of Computer Science, University of Wrocław, Poland. abraka@ii.uni.wroc.pl. Work done while the author was at MPII supported by Marie Curie Doctoral Fellowship.

allocation of projects to students and the allocation of probationary posts to trainee teachers, for instance.

The question arises as to how we might define a notion of optimality that will allow us to compare matchings, and how we might efficiently find an optimal matching. In the case where preferences are expressed on both sides, we have the notion of various kinds of *stability* to describe optimal matchings. This is the domain of *stable matching* problems, which have been studied extensively [2, 4, 5, 7, 9, 11]. When preferences are expressed on one side only (only applicants have preferences over posts), a number of different kinds of optimality can be defined. Here we study the notion of *rank-maximal* matchings, introduced in [8].

Definition 1 *Let r be the largest rank that an applicant uses to rank any post. The* signature *of a matching M is defined to be the r-tuple $(x_1,...,x_r)$ where for each $1 \leq i \leq r$, x_i is the number of applicants who are matched in M with one of their i-th choice posts.*

As a matter of convenience, we abbreviate a signature $(x_1,...,x_r)$ by $(x_1,...,x_d)$ if $x_d > 0$ and $x_i = 0$ for $i = d+1,...,r$. We use $\prec$ to denote the lexicographic order on signatures: $(x_1,...,x_r) \prec (y_1,...,y_r)$ if $x_i = y_i$ for $1 \leq i < k$ and $x_k < y_k$, for some k. Denote by $\mathscr{M}$ the set of all matchings of $\mathscr{A}$ to $\mathscr{P}$.

Definition 2 *A matching that has the maximum signature under this ordering is a* rank-maximal *matching. Alternately, and equivalently, define $\mathscr{M}_1$ to be the subset of $\mathscr{M}$, in which the maximum possible number of applicants are matched to their first choice post. For $i = 2,3,...,r$ define $\mathscr{M}_i$ to be the subset of $\mathscr{M}_{i-1}$ in which the maximum possible number of applicants are matched to their ith choice post. A matching that belongs to $\mathscr{M}_r$ is a* rank-maximal *matching.*

For a given problem instance, there might be more than one rank-maximal matching, but all rank-maximal matchings must have the same cardinality and the same signature.

It is easy to see that a simple greedy algorithm, in which we assign the maximum number of applicants to their first choice post, then the maximum number to their second choice post, and so on, is by no means guaranteed to lead to a rank-maximal matching.

Irving [8] considered the problem of computing a rank-maximal matching in instances where the preference list for any applicant $a \in \mathscr{A}$ is strictly ordered, that is, there are no ties in a's list. The running time of the algorithm in [8] is $O(\sum_{k=1}^{C} k^2(x_k+1)(n+s_k^2))$, where $(x_1,...,x_C)$ is the signature of a rank-maximal matching and $s_k = x_1 + \cdots + x_k$. The worst case complexity of this algorithm in the case where all preference lists have length at most c, is $O(c^2n^3)$; observe that for $(x_1,\ldots,x_c) = (1,1,\ldots,1,n-c+1)$, the last term in the sum is $O(c^2n^3)$.

A rank-maximal matching can also be found by transforming the input to an instance of the classical maximum weight bipartite matching problem. This involves allocating a suitably steeply decreasing sequence of weights to the edges. For instance, giving weight n^{r-i} to rank i edges. The scaling algorithm of Gabow and Tarjan [1] solves such instances with $O(\sqrt{n}m\log(n^r))$ arithmetic operations (additions and comparisons) on numbers bounded by n^r. Under the standard assumption that numbers of magnitude $O(n)$ can be handled in constant time and constant space, the resulting running time is $O(r^2\sqrt{n}m\log n)$ and the space requirement is $O(rm)$.

We give a simple combinatorial algorithm for constructing a rank-maximal matching. It runs in time $O(\min(n+C, C\sqrt{n})m)$, where $C \leq r$ is the maximal rank used in an optimal solution. Our algorithm runs in iterations. Let the edge set of G be $\mathscr{E} = \mathscr{E}_1 \cup \ldots \cup \mathscr{E}_r$, where $\mathscr{E}_i$ is the set of edges of rank i. In iteration i, our algorithm constructs a rank-maximal matching M_i in the graph $G_i = (\mathscr{A} \cup \mathscr{P}, \mathscr{E}_1 \cup \ldots \cup \mathscr{E}_i)$. The algorithm constructs M_{i+1} by computing a maximum matching (by augmenting M_i) in a suitable subgraph of G_{i+1}. Note that a rank-maximal matching is very different from a maximum cardinality matching but we modify the graph G_{i+1} so as to transform the rank-maximal matching problem to that of computing a maximum matching.

In Section 2 we describe our algorithm and analyze it. Section 3 contains some concluding remarks.

2 A Combinatorial Algorithm

In this section we present a combinatorial algorithm for computing a rank-maximal matching in a bipartite graph $G = (\mathscr{A} \cup \mathscr{P}, \mathscr{E})$. Before presenting the algorithm, let us examine the structure of the problem and build some intuition.

2.1 Some intuition

Recall that the edge set is $\mathscr{E} = \mathscr{E}_1 \cup \mathscr{E}_2 \ldots \cup \mathscr{E}_r$. For the present, let us assume that for every i and every $a \in \mathscr{A}$, $\mathscr{E}_i$ contains exactly one edge incident to a. First, we notice that in this case we can tell at once how many edges from $\mathscr{E}_1$ belong to a rank-maximal matching as well as which vertices from $\mathscr{P}$ are matched by them, namely exactly the vertices in $\mathscr{P}$ incident to an edge in $\mathscr{E}_1$. Let us denote this subset of $\mathscr{P}$ by $\mathscr{P}_1$. If some vertex $p \in \mathscr{P}_1$ is connected through $\mathscr{E}_1$ edges to more than one vertex $a \in A$, then we know only that, in a rank-maximal matching, one such vertex a must be matched with p, but we do not know which one. Nevertheless, without prejudice to the final outcome, we can delete from the graph all the edges of rank greater than one that are incident to vertices belonging to $\mathscr{P}_1$.

We observe that if we match all the vertices in $\mathscr{P}_1$ through $\mathscr{E}_1$ edges arbitrarily, delete edges of rank greater than one that are incident to vertices in $\mathscr{P}_1$ and then extend this matching along augmenting paths, then the number of $\mathscr{E}_1$ edges in the matching will not change.

What can we say about $\mathscr{E}_2$ edges? Here we may be uncertain as to which

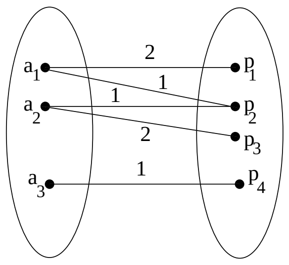

Figure 1: One of p_1, p_3 can be matched with a rank 2 edge.

vertices of $\mathscr{P}$ should be matched through $\mathscr{E}_2$ edges in a rank-maximal matching. This is the case, for example, in Figure 1, which illustrates part of the graph G_2 for a particular instance. We know that p_2 must be matched through an $\mathscr{E}_1$ edge but we do not know a priori which of p_1, p_3 should be matched with an $\mathscr{E}_2$ edge. However, vertices a_1 and a_2 must be matched by edges of rank at most two in a rank-maximal matching. Hence, we can delete all edges incident upon a_1, a_2 that are of rank greater than two. But do deletions of this kind suffice?

Our objective is to delete all edges that we know will never belong to a rank-maximal matching in order to transform the rank-maximal matching problem to a maximum matching problem in the reduced graph. We will show that the edges to be deleted can be determined using some well-known concepts and facts from matching theory. We can also drop the assumption that for every i, $\mathscr{E}_i$ contains exactly one edge incident to any $a \in A$.

2.2 Even, odd and unreachable vertices

Let M be a maximum matching in a bipartite graph G'. We recall that the vertex set of G' can be partitioned, relative to M, into three disjoint sets: E, O, and U. Nodes in E, O, and U are called *even, odd,* and *unreachable*, respectively [3]. E (O) consists of the nodes that can be reached in G' from a free node by an even (odd) length alternating path (with respect to M), and U consists of the nodes that cannot be reached from a free node by any alternating path. In Figure 1, $\{(a_1, p_1), (a_2, p_2), (a_3, p_4)\}$ is a maximum matching. It is easy to verify that $E = \{p_1, p_2, p_3\}$, $O = \{a_1, a_2\}$ and $U = \{a_3, p_4\}$. For vertex sets A and B, we call an edge connecting a vertex in A with a vertex in B an AB edge. Thus an OO edge is an edge connecting two vertices in O. The following lemma is well-known in matching theory. We include its proof for completeness.

Lemma 1 *Let M be a maximum matching in G' and let E, O and U be defined as above.*

1. *The sets E, O and U are pairwise disjoint.*

2. *Let N be any maximum matching in G'.*

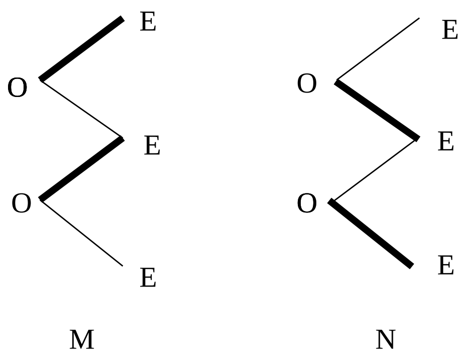

Figure 2: Matching edges are shown in bold. The left side shows a maximum matching M and an alternating path with respect to M starting at a free node. Also the partition of the vertex set is indicated. Switching edges in the path yields the maximum matching N. Observe that the partition of the vertex set does not change and that the vertices in O remain matched.

(a) N defines the same sets E, O and U.

(b) N contains only UU and OE edges.

(c) Every vertex in O and every vertex in U is matched by N.

(d) Its cardinality is equal to $|O| + |U|/2$.

3. There is no EU edge and no EE edge in G'.

Proof: 1. By definition, U is disjoint from O and E. It remains to show that O and E are disjoint. Assume a node v is reachable by an even length alternating path from free node a and by an odd length alternating path from free node b. (Note that $a \neq b$ since G' is a bipartite graph.) Then v is on the same side as a and the composition of the paths is an augmenting path from a to b. Thus M is not maximum, a contradiction.

2. Consider any maximum matching N. Then $M \oplus N$ consists of a set of alternating cycles and alternating paths, and each such cycle and path has even length. This is obvious for the cycles. For the paths it follows from the maximality of M and N. An alternating path containing more edges of N than edges of M would be augmenting with respect to M and hence contradict the maximality of M. Similarly, an alternating path containing more edges in M than edges in N would contradict the maximality of N. Using these paths and cycles to switch between M and N does not change the (even/odd/unreachable) status of any node, see Figure 2, and leaves the odd and the unreachable nodes matched. Thus the partition into sets E, O and U is independent of the particular maximum matching used to define it.

If a matched node is reachable by an even (odd) length alternating path from a free node, its mate is reachable by an odd (even) length alternating path from the same free node. Thus all edges in N are either UU or OE edges. Also, every node

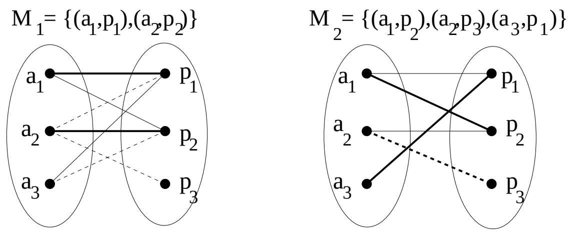

Figure 3: A simple example.

in U must be matched by N (otherwise it would be reachable by a path of length zero) and every node in O must be matched by N (since an odd length alternating path starting and ending at a free node is an augmenting path). Thus the cardinality of N is $|O| + |U|/2$.

3. Nodes in E are reachable by even length alternating paths. Such paths end in a matching edge. Also, EU edges are non-matching by part 2(b) and hence any such edge could be used to extend the alternating path, a contradiction of the definition of U.

Finally, since nodes in E are reachable by alternating paths ending in a matching edge, if there were an edge between two nodes of E, then it would be a non-matching edge and we could use it to construct an augmenting path. This would contradict the maximality of M. ▮

2.3 More intuition

The above facts formalize the ideas that we developed at the beginning of this section. Consider the edge set $\mathscr{E} = \mathscr{E}_1 \cup \mathscr{E}_2$. We first determine a maximum matching M_1 in $G_1 = (\mathscr{A} \cup \mathscr{P}, \mathscr{E}_1)$ and identify the sets of odd, even and unreachable vertices, O_1, E_1, and U_1. We then remove all rank two edges incident upon vertices in $O_1 \cup U_1$, because such vertices must be matched by edges in $\mathscr{E}_1$ in any matching using a maximum number of rank one edges (Part 2(c) of Lemma 1). We also remove all rank one O_1O_1 and O_1U_1 edges since no such edge appears in a matching that contains a maximum number of rank one edges (Part 2(b) of Lemma 1). Then we augment M_1 to obtain a matching M_2. Since M_2 is obtained by augmenting M_1, vertices matched in M_1 are still matched in M_2. So, vertices in O_1 and U_1 are still matched. By virtue of the edges that we removed, we know that each vertex in O_1 has to be matched by a rank one O_1E_1 edge and each vertex in U_1 is matched by a rank one U_1U_1 edge. Hence, M_2 has at least $|O_1| + |U_1|/2$ vertices matched by rank one edges. So, in this way we preserve the number of vertices that are matched by rank one edges. If we prove that no edge that we removed can ever occur in a rank-maximal matching and since M_2 is a maximum matching in the remaining graph, then we can see that M_2 is indeed a rank-maximal matching in $(\mathscr{A} \cup \mathscr{P}, \mathscr{E}_1 \cup \mathscr{E}_2)$.

Consider the simple example shown in Figure 3, where the solid lines indicate rank one edges and the dashed lines indicate rank two edges. $M_1 = \{(a_1, p_1), (a_2, p_2)\}$ is a maximum matching in the graph G_1 consisting of only rank one edges, and $O_1 = \{p_1, p_2\}, E_1 = \{a_1, a_2, a_3, p_3\}, U_1 = \emptyset$. Now we remove the edges (a_2, p_1) and (a_3, p_2) since these are rank two edges incident on vertices in O_1. Then we augment M_1 to obtain M_2, which consists of two edges of $\mathscr{E}_1$ and one edge of $\mathscr{E}_2$. If we had augmented M_1 without removing the rank two edge (a_3, p_2), then we would have ended up with a maximum matching containing one edge of $\mathscr{E}_1$ and two edges of $\mathscr{E}_2$ - not a rank-maximal matching.

2.4 The algorithm

We now present our algorithm for constructing a rank-maximal matching M. Let $G_i = (\mathscr{A} \cup \mathscr{P}, \mathscr{E}_1 \cup \ldots \cup \mathscr{E}_i)$. We start with $G'_1 = G_1$, and M_1 any maximum matching in G'_1.

For $i = 1$ to $r-1$ do the following steps, and output M_r.

1. Partition the nodes of $\mathscr{A} \cup \mathscr{P}$ into three disjoint sets: E_i, O_i, and U_i. E_i and O_i consist of the nodes that can be reached in G'_i from a free node by an even or odd length alternating path (with respect to M_i) respectively, and U_i contains the nodes that cannot be reached from a free node by an alternating path.

2. Delete all edges incident to a node in $O_i \cup U_i$ from $\mathscr{E}_j$ for all $j > i$. $O_i \cup U_i$ consists of the nodes that are matched by every maximum matching of G'_i. Delete all O_iO_i and O_iU_i edges from G'_i. These are the edges that are not used by any maximum matching of G'_i. Add the edges in $\mathscr{E}_{i+1}$ to G'_i, and call the resulting graph G'_{i+1}.

3. Determine a maximum matching M_{i+1} in G'_{i+1} by augmenting M_i. (Note that M_i is still contained in G'_{i+1}.)

Observe that the algorithm maintains the invariant that M_{i+1} is a maximum matching in G'_{i+1}. We will show in the proof of correctness that the following invariants are also maintained:

- every rank-maximal matching in G_{i+1} has all of its edges in G'_{i+1}
- M_{i+1} is a rank-maximal matching in G_{i+1}.

Suppose $(s_1, s_2, \ldots, s_i, \ldots)$ is the signature of a rank-maximal matching. We will show that M_i has signature $(s_1, s_2, \ldots, s_i)$.

2.5 The proof of correctness

We start with the following lemma, which proves that the edges that are deleted during phase $i+1$ in our algorithm do not belong to any rank-maximal matching of G_{i+1}, provided that we maintain our invariants until the end of phase i.

Lemma 2 *Suppose that every rank-maximal matching of G_i is a maximum matching of G'_i. Then every rank-maximal matching of G_{i+1} is contained in G'_{i+1}.*

Proof: We need to show that the edges that we removed in the $(i+1)$th phase of the algorithm do not belong to any rank-maximal matching of G_{i+1}.

Let N_{i+1} be any rank-maximal matching of G_{i+1}. It has signature $(s_1, s_2, ..., s_{i+1})$. $N_i = N_{i+1} \cap \mathscr{E}_{\leq i}$ is a matching with signature $(s_1, s_2, ..., s_i)$ and is therefore a rank-maximal matching of G_i. So N_i is a maximum matching in G'_i by the assumption. By Lemma 1 parts 2(b) and (c), it has to pair the nodes in U_i and match all nodes in O_i with nodes in E_i.

Thus N_i does not use any O_iO_i or O_iU_i edge of G_i. Since N_{i+1} is a matching, it cannot use any such edge either, and moreover, it cannot use any edge of rank higher than i incident to some node in $O_i \cup U_i$. So N_{i+1} is contained in G'_{i+1}. ∎

In addition, the deletions guarantee that the number of edges of each smaller rank is preserved throughout the algorithm:

Lemma 3 *For every i, j ($j > i$), the number of edges of rank at most i is the same in M_i and M_j.*

Proof: Since M_j is obtained from M_i by successive augmentations, every vertex matched by M_i is also matched by M_j. Hence, all nodes in U_i and O_i are matched in M_j.

Since G'_j has no edges of rank greater than i incident to nodes in O_i and U_i, and no O_iO_i edge or O_iU_i edge of rank $\leq i$, M_j must pair the nodes in U_i and must match nodes in O_i with nodes in E_i. These edges have rank at most i. So, M_j has at least as many edges of rank $\leq i$ as M_i, and M_j cannot have more since all the edges of rank $\leq i$ in G'_j belong to G'_i and M_i is a maximum matching in G'_i. ∎

Now, we are ready to prove the correctness of our algorithm.

Theorem 1 *For every $1 \leq k \leq r$, the following statements hold:*
(i) *Every rank-maximal matching in G_k is a maximum matching in G'_k;*
(ii) *M_k is a rank-maximal matching in G_k.*

Proof: We prove this by induction on k. Since all edges in G_1 have the same rank, a rank-maximal matching in G_1 is the same as a maximum matching. Since M_1 is a maximum matching in G_1, both statements hold for $k = 1$.

Let us now prove these statements for $i+1$ assuming them to be true for i. Since M_i is a rank-maximal matching in G_i, its signature is $(s_1, \ldots, s_i)$. Suppose the signature of M_{i+1} is $(r_1, \ldots, r_i, r_{i+1})$. By Lemma 3, we know that, for every k ($1 \leq k \leq i$), $\sum_{j=1}^{k} s_i = \sum_{j=1}^{k} r_i$. So it follows that the signature of M_{i+1} is $(s_1, \ldots, s_i, r_{i+1})$ for some $r_{i+1} \leq s_{i+1}$.

By the induction hypothesis, every rank-maximal matching of G_i is a maximum matching of G'_i. Hence, by Lemma 2, any rank-maximal matching of G_{i+1} is contained in G'_{i+1}. Thus there is a matching of cardinality $s_1 + \ldots + s_i + s_{i+1}$ in G'_{i+1}. Since M_{i+1} is a maximum matching in G'_{i+1}, its cardinality is at least $s_1 + \ldots s_i + s_{i+1}$. Thus $r_{i+1} = s_{i+1}$.

Hence M_{i+1} is a rank-maximal matching in G_{i+1}. It is now easy to show that every rank-maximal matching of G_{i+1} is a maximum matching of G'_{i+1}. Let N_{i+1} be any rank-maximal matching of G_{i+1}. By the induction hypothesis and Lemma 2, we know that N_{i+1} is contained in G'_{i+1}. Furthermore, N_{i+1} has cardinality $s_1 + s_2 + \ldots + s_{i+1}$, which is equal to the cardinality of M_{i+1}, which is a maximum matching of G'_{i+1}. Hence, N_{i+1} is also a maximum matching of G'_{i+1}. This completes the proof of the lemma. ▮

2.6 The running time of the algorithm

Theorem 2 *A rank-maximal matching can be computed in $O(\min(C\sqrt{n},\ n+C) \cdot m)$ time, where C is the maximal rank of an edge in an optimal solution*

Proof: Consider a fixed iteration i. We first determine the partition of the node set into E_i, O_i, and U_i. This can be done in $O(m)$ time by growing a Hungarian forest with respect to M_i. Then we delete appropriate edges from the edge set, which again takes $O(m)$ time. We next compute M_{i+1} from M_i by augmenting along augmenting paths. Using the algorithm of Hopcroft and Karp [6], this takes time $O(\min(\sqrt{n},\ |M_{i+1}| - |M_i| + 1) \cdot m)$. The number of iterations is r and hence the overall running time is $O(\min(r\sqrt{n}, n + r) \cdot m)$.

We next show how to replace r by C. At the beginning of each iteration, say iteration i, we first check whether M_i is already a maximum matching in G'_r, where G'_r denotes the graph consisting of all edges, of all ranks, that are still present at the beginning of phase i. This takes time $O(m)$. If M_i is a maximum matching in G'_r, then we stop. Otherwise, there is an edge of rank greater than i whose addition increases the cardinality of the maximum matching, and so $C > i$. In this case, we continue as described above. In this way, only C iterations are executed. ▮

3 Conclusions

We presented an algorithm to compute a rank-maximal matching in a bipartite graph where each edge has a rank. The algorithm has running time $O(C\sqrt{n}m)$. There are several natural variations of the rank-maximal matching problem.

Posts may have capacities: a post p of capacity $c(p)$ can be matched with up to $c(p)$ applicants. We can solve this problem by making $c(p)$ copies of p with each of them adjacent to the same neighbors as p and then solving the rank-maximal matching problem in the resulting graph.

Instead of asking for the matching maximizing the signature $(s_1, s_2, \ldots, s_r)$ we might be interested in the maximal cardinality matching maximizing $(s_1, s_2, \ldots, s_r)$ or minimizing $(s_r, s_{r-1}, \ldots, s_1)$. We do not know whether our algorithm generalizes to these problems. However, the reduction to weighted matchings still works and yields algorithms with running time $O(r^2\sqrt{n}m\log n)$ and space requirement $O(rm)$. In [10], the running time is improved to $O(r\sqrt{n}m\log n)$ and the space requirement is reduced to $O(m)$.

References

[1] H. N. Gabow and R.E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal of Computing*, 18:1013–1036, 1989.

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

[3] R. L. Graham, M. Grötschel, and L. Lovász, editors. *The Handbook of Combinatorics*, chapter 3, Matchings and Extensions, pages 179–232. North Holland, 1995.

[4] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[5] M. M. Halldórsson, K. Iwama, S. Miyasaki, and H. Yanagisawa. Improved approximation of the stable marriage problem. In *ESA 2003*, volume 2832 of *Lecture Notes in Computer Science*, pages 266–277, 2003.

[6] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

[7] R. W. Irving. Matching medical students to pairs of hospitals: a new variation on a well-known theme. In *Proceedings of the Sixth European Symposium on Algorithms*, volume 1461 of *Lecture Notes in Computer Science*, pages 381–392, 1998.

[8] R. W. Irving. Greedy matchings. Technical Report TR-2003-136, University of Glasgow, April 2003.

[9] R. W. Irving, D. F. Manlove, and S. Scott. Strong stability in the hospitals-residents problem. In *Proceedings of the STACS*, volume 2607 of *Lecture Notes in Computer Science*, pages 439–450, 2003.

[10] K. Mehlhorn and D. Michail. Network problems with non-polynomial weights and applications. `www.mpi-sb.mpg.de/~mehlhorn/ftp/HugeWeights.ps`.

[11] A. E. Roth. The evaluation of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984.