Reducing Rank-Maximal to Maximum Weight Matching

Dimitrios Michail [a,*]

[a] *Max-Planck-Institut für Informatik, Saarbrücken, Germany*

Abstract

Given a bipartite graph $G(V, E)$, $V = A \dot\cup B$ where $|V| = n, |E| = m$ and a partition of the edge set into $r \leq m$ disjoint subsets $E = E_1 \dot\cup E_2 \dot\cup \ldots \dot\cup E_r$, which are called ranks, the *rank-maximal matching* problem is to find a matching M of G such that $|M \cap E_1|$ is maximized and given that $|M \cap E_1|$ is maximized, $|M \cap E_2|$ is also maximized, and so on. Such a problem arises as an optimization criteria over a possible assignment of a set of applicants to a set of posts. The matching represents the assignment and the ranks on the edges correspond to a ranking of the posts submitted by the applicants.

The rank-maximal matching problem and several other optimization variants, e.g. *fair matching* and *maximum cardinality rank-maximal matching*, can be solved by a reduction to the weight matching problem in time $O(r\sqrt{n}m \log n)$. Recently, Irving et al. developed a combinatorial approach which improves the running time for the rank-maximal matching problem to $O(\min(n + r, r\sqrt{n})m)$. They raised the open questions on (a) whether such a running time can be achieved by the weight matching reduction and (b) whether such a running time can be achieved for the other variants of the problem.

In this work we show how the reduction to the weight matching problem can also be used to achieve the same running time. Our algorithm is simpler and more intuitive.

Key words: bipartite graph, matching, preference lists, rank maximal, weighted matching

* Corresponding address: Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, Tel: 0049 681 9325 128, Fax: 0049 681 9325 199

Email address: michail@mpi-inf.mpg.de (Dimitrios Michail).

1 Introduction

Consider a bipartite graph $G(V, E)$, $V = A \dot\cup B$, where $|V| = n, |E| = m$ and a partition of the edge set into $r \leq m$ disjoint subsets $E = E_1 \dot\cup E_2 \dot\cup \ldots \dot\cup E_r$. Each set of edges E_i is denoted as the edges of rank i. To avoid triviality we assume that $n > 3$ and $m = \Omega(n)$. A matching M is called rank-maximal if it maximizes $|M \cap E_1|$, and given that $|M \cap E_1|$ is maximized, it maximizes $|M \cap E_2|$, and so on. Note that such a matching is not necessarily of maximum cardinality. The problem arises as an optimization criteria when assigning a set of applicants to a set of posts. In this setting we call the nodes in A *applicants* and the nodes in B *posts*. If $(a, b) \in E_i$ and $(a, b') \in E_j$ with $i < j$, we say that a prefers b to b'. If $i = j$, we say that a is *indifferent* between b and b'. This ordering of posts adjacent to a is called a's preference list. A matching M corresponds to the assignment and the ranks on the edges correspond to a ranking of the posts submitted by the applicants.

Such a bipartite matching problem with a graded edge set is well studied in the economics literature, see for example [1–3]. It models important real-world problems, including the allocation of graduates to training positions [4]. Using the problem setup considered in this paper, various other definitions of optimality have been considered. For example, a matching M is *Pareto optimal* [5,1,2] if there is no matching M' such that (i) some applicant prefers M' to M, and (ii) no applicant prefers M to M'. Another important definition of optimality is *Popular matchings* [6]. A matching M is popular if and only if there is no matching M' that is more popular that M. A matching M' is more popular than M if a majority of applicants in M would prefer to switch to M'. It is important to note that popular matchings may not always exist. Finally, we mention *maximum-utility* matchings, which maximize $\sum_{(a,b)\in M} u_{a,b}$, where $u_{a,b}$ is the utility of allocating applicant a to post b. Maximum-utility matchings can be found using an obvious transformation to the maximum-weight matching problem. All the above problems belong to the class of one-sided preference list problems. When preference lists are expressed from both sides of the partition we have the well-known problem of *stable marriage*.

The rank-maximal matching problem can be solved by using the well-studied maximum weight matching problem. Given an edge e of rank i let its weight be $2^{\lceil \log n \rceil (r-i)}$. Since any matching M has less that n edges, it follows that no collection of edges of rank at least i can replace an edge of rank $i-1$ and therefore a maximum weight matching in this instance corresponds to a rank-maximal matching. The maximum weight matching problem has been previously studied and efficient algorithms exist. The fastest algorithms have running time $O(n(m + n \log n))$ [7] and $O(\sqrt{n}m \log nC)$ [8] where C is the maximum weight in an instance. The first is strongly- while the second weakly-polynomial. Simply applying the aforementioned algorithms to the rank-maximal matching

problem does not result in efficient algorithms, either in time nor in space requirements. The problem is that the edge weights are as large as n^r, which is non-polynomial in the input size. Both algorithms assume that arithmetic operations between numbers which are in $O(C)$ can be performed in constant time. This is not true in this case, where arithmetic on numbers in $O(n^r)$ takes time $\Omega(r)$. Hence, the running times are $O(rn(m + n \log n))$ and $O(r^2\sqrt{n}m \log n)$ respectively, both using $O(rn)$ space. It is known [9], however, that the scaling algorithm for the weighted matching problem can be implemented such that all arithmetic is performed on numbers with $O(\log n)$ bits, independent of the edge weights. In this case the running time improves to $O(r\sqrt{n}m \log n)$.

Given the partition of the edge set $E = E_1 \dot\cup E_2 \dot\cup \ldots \dot\cup E_r$ we can also ask for matchings satisfying different criteria. The *maximum cardinality rank-maximal* matching problem asks for a rank-maximal matching with maximum cardinality. Similarly the *fair* matching problem asks for a matching of maximum cardinality such that the minimum number of edges of rank r are used, and given that the minimum number of edges of rank $r-1$ are used, and so on. The fastest method, so far, to solve these problems is by using a similar reduction to the weighted matching problem. These problems have been recently considered by Abraham et al. [10] in the context of rental markets. Finally let us mention that combinations of different criteria are also possible. For example the *fair-popular* matchings problem [11] asks for a popular matching which is the most fair among popular matchings.

In [12,13] the authors present a combinatorial algorithm which solves the rank-maximal matching problem in $O(min(n + r, r\sqrt{n})m)$ time using linear space. The algorithm identifies edges which cannot be part of a rank-maximal matching and deletes them. This approach, however, does not seem to generalize to the maximum cardinality rank-maximal or the fair matching problem. In an attempt to close the gap between the rank-maximal matching and its variants, we present an algorithm which solves the rank-maximal matching problem in the same running time and space as [12]. The main difference is than our algorithm is based on the weight matching reduction. We believe that the new algorithm is simpler and more intuitive, since the weighted matching approach seems as the most natural way to approach the problem. Morever, it should be a better starting point in designing faster algorithms for the other variants of the rank-maximal matching problem.

2 Preliminaries

Let $\pi : V \mapsto \mathbb{Z}_{\geq 0}$ be a *potential* function defined on the vertices of G. For an edge $e = (v, w) \in E$ denote its weight by $c(e)$ and define its *reduced weight*

with respect to π as $\bar{c}(e) = \pi(v) + \pi(w) - c(e)$. Moreover call such an edge *tight* if $\bar{c}(e) = 0$. We say that π is a feasible potential function for c if $\bar{c}(e) \geq 0$ for all edges $e \in E$. We say that a feasible potential function π is *optimal* if there is a matching M in G such that $\bar{c}(e) = 0$ for each $e \in M$ and $\pi(v) = 0$ for all $v \in V$ which are free in M. Define also $\Pi = \sum_{v \in V} \pi(v)$. Note that any matching M has

$$c(M) = \sum_{e=(v,w) \in M} c(e) \leq \sum_{e=(v,w) \in M} \pi(v) + \pi(w) \leq \sum_{v \in V} \pi(v) = \Pi$$

Moreover, for an optimal potential function π and the corresponding matching M, $c(M) = \Pi$, since any free vertex has zero potential.

Our algorithm uses a decomposition theorem by Kao et al. [14]. For an integer $h \in [1, C]$, where C is the maximum weight of an edge, divide G into two lighter subgraphs G_h and G'_h as follows:

- G_h is formed by edges $(u, v) \in G$ such that $c(e) \in [C - h + 1, C]$. An edge $e \in G_h$ has weight $c_h(e) = c(e) - (C - h)$.
- Let π_h be an optimal potential function for G_h. An edge $e = (u, v) \in G$ belongs to G'_h if $\pi_h(u) + \pi_h(v) - c(e) < 0$. In that case edge e has weight $c'_h(e) = c(e) - \pi_h(u) - \pi_h(v)$ in G'_h.

Theorem 1 ([14]) *Consider G, G_h and G'_h as defined above and let* $\mathrm{mwm}(G)$ *denote the weight of a maximum weight matching in G. Then* $\mathrm{mwm}(G) = \mathrm{mwm}(G_h) + \mathrm{mwm}(G'_h)$.

If π_h and π'_h are optimal potential functions for G_h and G'_h respectively, then $\pi_h + \pi'_h$ is an optimal potential function for G. This follows directly by Theorem 1 and the feasibility of $\pi_h + \pi'_h$ w.r.t the original weight function. See [14] for more details.

3 The decomposition

Consider an instance of the rank-maximal matching problem and the reduced instance of the weight matching problem. The edges of G have weights of the form [1] $1, n, n^2, \ldots, n^{r-1}$. Using Theorem 1 we decompose the problem and solve it recursively. The base case of the recursion is a maximum cardinality matching computation.

[1] We say that a graph G has edges with weights of the form $1, n, n^2, \ldots, n^{r-1}$ to denote that all edges of G can be assigned to one of these categories based on their weight; multiple edges can have the same weight.

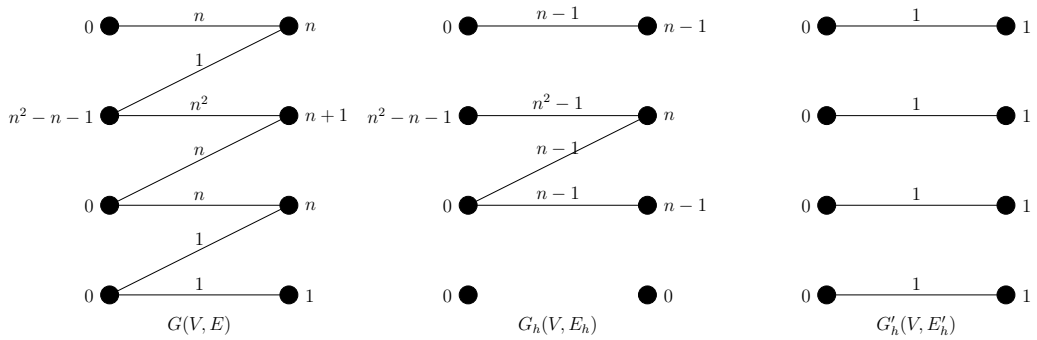

Fig. 1. Example of an instance of the rank-maximal matching problem reduced to maximum weight matching and the two graphs resulting from the decomposition. The numbers next to the edges represent the costs. The numbers next to the vertices represent an optimal potential function.

Choose $h = n^{r-1} - 1$. Then G_h contains the edges of G with weight in the range $[2, n^{r-1}]$. Each edge in G_h has weight $c_h(e) = c(e) - 1$. Thus, graph G_h contains all edges of G with rank at most $r - 1$ and these edges have weights $n - 1, n^2 - 1, \ldots, n^{r-1} - 1$. Graph G'_h is much simpler. Assuming that π_h is an optimal potential function for G_h, G'_h contains only the edges of G with negative reduced weight w.r.t π_h. Such edges fall into two categories:

- Edges $e = (u, w) \in G$ where $c(e) = 1$ and $\pi_h(u) + \pi_h(w) = 0$. Such edges have cost 1 in G'_h.
- Edges $e = (u, w) \in G$ where $c(e) > 1$ and $\pi_h(u) + \pi_h(w) - c(e) < 0$. Due to the feasibility of π_h in G_h,

$$\pi_h(u) + \pi_h(w) - c_h(e) \geq 0$$

  where $c_h(e) = c(e) - 1$. We conclude that $\pi_h(u) + \pi_h(w) - c(e) \geq -1$ and therefore all such edges also have cost 1 in G'_h. These edges are exactly the ones in G_h which are tight w.r.t. π_h.

The decomposition results in two subproblems (see Figure 1 for an example). The subproblem in G'_h is a maximum cardinality matching computation, since all edges have weight 1. On the other hand graph G_h has edges with weights of the form $n - 1, n^2 - 1, \ldots, n^{r-1} - 1$. We are going to show in Section 4 that an optimal potential function for these weights is also optimal for the weights $1, n, n^2, \ldots, n^{r-2}$. Thus, the subproblem in G_h is a rank-maximal matching computation with $r - 1$ ranks. This can be solved recursively in time $T(r - 1)$ where $T(r)$ is the time required to solve an instance with r different ranks.

Extra care is required in order to watch out for the cost of arithmetic operations during the algorithm, since the potential function π_h can take values up to $O(n^r)$. In this respect consider the following representation for an optimal potential function π:

- A set of nodes V^0 containing all nodes $v \in V$ s.t $\pi(v) = 0$.
- A set of edges E^0 containing all edges $e = (u, w) \in E$ which are tight w.r.t π, i.e., $\bar{c}(e) = \pi(u) + \pi(w) - c(e) = 0$.

The above two sets can guide the construction of a matching M in $O(\sqrt{n}m)$ time such that any matched edge is tight and every free vertex has zero potential. The above imply that such a matching has the same cost as our optimal potential function and, therefore, is itself optimal. In the algorithm we will manipulate these two sets and maintain the invariant that all such tuples will correspond to the representation of some optimal potential function.

4 Fewer ranks

Let G be a graph with edge weights $1, n, n^2, \ldots, n^{r-2}$ and let V^0, E^0 be two sets representing an optimal potential function. In this section we show that the same sets are a solution for the edge weights $n-1, n^2-1, \ldots, n^{r-1}-1$. More precisely, there exists an optimal potential function such that these two sets are its representation.

Assume that we have solved the subproblem with edge costs $1, n, n^2, \ldots, n^{r-2}$ in time $T(r-1)$ and we have an optimal potential function π represented by V^0 and E^0. In order to obtain an optimal potential function for the edge costs $n-1, n^2-1, \ldots, n^{r-1}-1$ the first step is to obtain an optimal potential function for the edge costs $n, n^2, \ldots, n^{r-1}$. The following lemma is part of the folklore.

Lemma 2 *Consider a graph G with edge costs $1, n, n^2, \ldots, n^{r-2}$. Let M be a maximum weight matching of G and $\pi : V \mapsto \mathbb{Z}_{\geq 0}$ be a potential function proving its optimality. Then the potential function $n\pi$ proves the optimality of M for G with edge costs $n, n^2, n^3, \ldots, n^{r-1}$.*

Let π' be the potential function obtained by multiplying π with n in Lemma 2. From the feasibility of π, the definition of π' and the integrality of the potential functions we also get the following corollary.

Corollary 3 *For any node $v \in V$ either $\pi'(v) = 0$ or $\pi'(v) \geq n$. Moreover, for any edge $e \in E$ either $\bar{c}(e) = \pi'(v) + \pi'(w) - c(e) = 0$ or $\bar{c}(e) \geq n$.*

The same sets V^0 and E^0 are a representation of the new optimal potential function π' for the new weights. The next step is the construction of a potential function π'' which will be optimal for the weights $n-1, n^2-1, \ldots, n^{r-1}-1$. Algorithm 1 constructs such a potential function. We will need the following definition.

Input: optimal potential function π' for edge costs $n, n^2, \ldots, n^{r-1}$
Output: optimal potential function π'' for edge costs
$n-1, n^2-1, \ldots, n^{r-1}-1$

while $G_=$ has a vertex v with $\pi'(v) = 0$
let A_v and B_v be the vertices reachable from v in $G_=$ by even and odd paths respectively
for $w \in A_v$ set $\pi''(w) = \pi'(w)$
for $w \in B_v$ set $\pi''(w) = \pi'(w) - 1$
delete A_v and B_v
endwhile
if $G_=$ is not empty
let $A \dot{\cup} B$ be the remaining vertex set of $G_=$
for each $w \in A$, set $\pi''(w) = \pi'(w)$
for each $w \in B$, set $\pi''(w) = \pi'(w) - 1$
endif

Algorithm 1. Construct optimal potential function

Definition 4 (equality subgraph) *For a graph $G(V, E)$ with edge costs $c : E \mapsto \mathbb{Z}_{>0}$ and a potential function $\pi : V \mapsto \mathbb{Z}_{\geq 0}$, let the equality subgraph $G_=(V, E_=)$ be the graph with edge set $E_= = \{e = (u, v) \in E : \bar{c}(e) = \pi(u) + \pi(v) - c(e) = 0\}$.*

The following lemmata prove the correctness of Algorithm 1.

Lemma 5 *Let G be a graph with edge costs $n, n^2, \ldots, n^r$, $n > 1$ and $\pi : V \mapsto \mathbb{Z}_{\geq 0}$ be an optimal potential function for G. Let $G_=$ be the equality subgraph of G w.r.t π and $v \in V$ be a vertex with $\pi(v) = 0$. Then, there is no odd length path in $G_=$ starting from v which ends to a vertex w with $\pi(w) = 0$.*

PROOF. Assume otherwise and let p be an odd length path in $G_=$ from v to w such that $\pi(v) = \pi(w) = 0$ (see Figure 2). Since every edge on p has reduced cost zero we get the following

$$\sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}} n^i = \sum_{j \in \{\beta_1, \beta_2, \ldots, \beta_{k-1}\}} n^j \tag{1}$$

where $\alpha_i \geq 1, \beta_i \geq 1, n > 1$ and $2k - 1 < n$. Assume w.l.o.g that there are no i, j such that $n^{\alpha_i} = n^{\beta_j}$, otherwise remove both of them. At least one term will remain since the number of terms is odd. Both sides have at most $k < n$ terms and any term in one side needs at least n terms on the other side in order to cancel out. But there are not so many terms, a contradiction. □

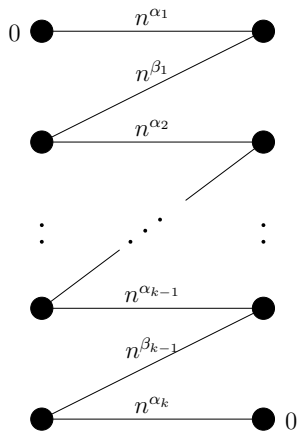

Fig. 2. Path p in equality subgraph $G_=$ for the proof of Lemma 5.

Lemma 6 *Algorithm 1 constructs an optimal potential function π'' for the edge costs $n-1, n^2-1, \ldots, n^{r-1}-1$ given an optimal potential function π' for the edge costs $n, n^2, \ldots, n^{r-1}$. The two potential functions have the same representation as sets V^0 and E^0.*

PROOF. By Corollary 3 every edge $e=(v,w)$ which does not belong to $G_=$ has $\bar{c}'(e) = \pi'(v) + \pi'(w) - c(e) \geq n$. Every potential can decrease by at most 1 during the algorithm and therefore $\bar{c}''(e) = \pi''(v) + \pi''(w) - (c(e)-1) \geq \pi'(v) - 1 + \pi'(w) - 1 - (c(e)-1) = \pi'(v) + \pi'(w) - c(e) - 1 \geq n - 1 > 0$ since $n > 1$. This means that all these edges are feasible and every such edge remains non-tight.

In the first part of the algorithm, by Lemma 5 all vertices v that get their potential decreased have $\pi'(v) > 0$. The same is true for the second part, since the second part is executed only if $G_=$ is non-empty and there is no vertex with zero potential. By Corollary 3 every such vertex has $\pi'(v) \geq n$ and therefore $\pi''(v) \geq n - 1 > 0$. Hence, the set of vertices with zero potential remain the same.

We will now examine the effect on $G_=$. For each edge e that is tight w.r.t π', exactly one of its endpoints get their potential reduced by 1 and therefore e remains tight in π'' w.r.t the cost function $c(e) - 1$.

Finally, let M be a maximum weight matching such that π' proves its optimality. Every edge $e \in M$ belongs to $G_=$ and every free vertex has zero potential. The potential function π'' also proves the optimality of M w.r.t the new cost function, since the sets V^0 and E^0 have not changed. Thus the resulting function is optimal. □

Input: graph G with edge partition $E_1, E_2, \ldots, E_r$
Output: sets V^0, E^0

if $r = 1$
  compute maximum matching M of G and optimal π
  based on π compute V^0 and E^0 and return them
else
  solve recursively instance for $G(V, E \setminus E_r)$ and $E_1, E_2, \ldots, E_{r-1}$
  let V^0, E^0 be the solution
  let $E^{\ddagger} = \{e = (v, u) \in E_r : \{v, u\} \in V^0\}$
  form unweighted $G'_h(V, E^{\ddagger} \cup E^0)$ and find optimal potential function π'_h in G'_h
  set $E^0 = \{e = (v, u) \in E^{\ddagger} \cup E^0 : \pi'_h(v) + \pi'_h(u) = 1\}$
  set $V^0 = \{v \in V^0 : \pi'_h(v) = 0\}$
  return V^0 and E^0
endif

Algorithm 2. Compute a rank-maximal matching

5 Combining the solutions

After solving the two subproblems we are left with the following:

- Graph G'_h and an optimal potential function π'_h which was obtained by a maximum cardinality matching computation, and
- graph G_h and sets V^0, E_0 representing an optimal potential function π_h.

Combining the two solutions requires to add up the two potential functions, π_h and π'_h. The addition will be performed implicitly by changing V^0 and E^0 based on the potential function π'_h. Updating V^0 requires checking for each $v \in V$ whether $\pi_h(v) = \pi'_h(v) = 0$ which can be done by checking whether $\pi'_h(v) = 0$ and $v \in V^0$. Updating E^0 is slightly more complicated.

- For an edge $e = (v, u)$ with $c(e) = 1$ in G, i.e. $e \in E_r$, we have to check whether $\pi_h(v) + \pi'_h(v) + \pi_h(u) + \pi'_h(u) = 1$. Recall from Section 4 that if a vertex $v \in V$ has $\pi_h(v) > 0$ then $\pi_h(v) > n-2$ and therefore it is enough to check (a) that $\pi'_h(v) + \pi'_h(u) = 1$ (an operation which takes constant time since π'_h is polynomially bounded) and (b) that $\pi_h(v) = \pi_h(u) = 0$ which can be done by checking whether $\{v, u\} \in V^0$.
- For the rest of the edges, we have to check whether $\pi_h(v) + \pi'_h(v) + \pi_h(u) + \pi'_h(u) = c(e)$. By the feasibility of π_h we know that $\pi_h(v) + \pi_h(u) \geq c(e) - 1$. Moreover, for an edge e s.t $\pi_h(v) + \pi_h(u) \neq c(e) - 1$ we know that $\pi_h(v) - 1 + \pi_h(u) - 1 - (c(e) - 1) \geq n - 1$ (in the worst case where both endpoints got their potential decreased by 1, when transforming to a new potential

function for edge weights which are reduced by 1), and hence any such edge cannot be tight.

We conclude that if an edge has $\pi_h(v) + \pi_h(u) = c(e) - 1$ and therefore belongs already to E^0, then it will remain tight if $\pi'_h(v) + \pi'_h(u) = 1$.

The final output of the algorithm is sets V^0 and E^0 which represent an optimal potential function. See Algorithm 2 for a succinct description.

From these sets a rank-maximal matching can be constructed by performing one maximum cardinality matching computation. This is done in the following way. Let $G_=(V, E^0)$ be the final equality subgraph. Create a new graph, $G^{\alpha\beta}$, containing two copies of $G_=$, $G^\alpha(V, E^\alpha)$ and $G^\beta(V, E^\beta)$. For a vertex $v \in V$, let v^α and v^β be the two copies in the new graph. Then, if $v \in V^0$ include the edge (v^α, v^β). Finally find a maximum cardinality matching M in $G^{\alpha\beta}$. The matching $M \cap E^\alpha$ is then a maximum weight matching in the original graph, as the following lemma proves.

Lemma 7 *The matching $M \cap E^\alpha$ constructed as above, is a maximum weight matching in G.*

PROOF. Let π be the optimal potential function for G, represented as the sets V^0 and E^0. We first argue that the graph $G^{\alpha\beta}$ has a perfect matching. Let M' be a maximum weight matching in G corresponding to π. Each edge in M' is tight w.r.t π and therefore belongs to E^0. Thus, M' is also a matching of $G_=$. Take two copies of this matching one for G^α and one for G^β. All remaining unmatched nodes in $G^{\alpha\beta}$ were also unmatched in M' and therefore have zero potential. We added the extra edges $v^\alpha v^\beta$ in case a vertex v had zero potential. Together with these edges a perfect matching for $G^{\alpha\beta}$ can be formed.

Let now M be a maximum cardinality matching of $G^{\alpha\beta}$. By the above M is a perfect matching. Consider any node v s.t $v \notin V^0$. There is no edge between v^α and v^β and therefore v^α must be matched inside G^α. Therefore, any node $v \notin V^0$ must be matched by $M \cap E^\alpha$. Hence,

$$\Pi = \sum_{v \in V} \pi(v) = \sum_{\substack{v \in V \\ v \notin V^0}} \pi(v) = \sum_{e \in M \cap E^\alpha} c(e) = c(M \cap E^\alpha)$$

and the lemma follows. □

In the case where $r = 1$ finding an optimal potential function is straightforward using a maximum cardinality matching. For example, the LEDA [15] library can return an optimal potential function alongside a maximum cardinality matching.

5.1 Running Time

Denote the algorithm's running time as $T(r)$ for an instance with r ranks. Moreover, assume that together with sets V^0, E^0 the recursive call returns a maximum cardinality matching M^0 of the graph induced by the edges in E^0. Then $T(r)$ consists of solving a subproblem of size $r-1$ recursively in $T(r-1)$, a maximum matching computation and the time taken to combine the two solutions. Finding a new maximum cardinality matching can be done in $O(min(\sqrt{n}, |M'| - |M^0| + 1)m)$ where M' is the new maximum matching. All the administrative work, like updating the various graphs and sets, does not dominate the running time. The recursion solves to $O(min(n+r, r\sqrt{n})m)$. The space requirement of the algorithm is linear.

5.2 Fewer phases

The number of phases can be reduced from r to r^*, where r^* is the largest rank used in an optimal solution for a particular instance. Assume we are at the start of phase i. We have sets V^0 and E^0 corresponding to edges of ranks $i-1, \ldots, 1$. These edges now have weights $n-1, \ldots, n^i - 1$. We can check whether this matching is already optimal by assuming that all remaining edges are of rank i, and therefore solving our initial problem with edge weights $n^{r-i}, \ldots, n^{r-i}, n^{r-i+1}, \ldots, n^{r-1}$ or equivalently for $1, \ldots, 1, n, \ldots, n^{i-1}$. This way we simply boost the importance of less important edges in order to check whether they would be used or not. Solving this instance can be easily done since the only change is in graph G'_h by including more edges with weight 1. To summarize, in phase i we first form G'_h by including the appropriate edges from edges $e \in E_i \cup E_{i+1} \cup \cdots \cup E_r$ and try to find a maximum weight matching in G'_h. If the resulting matching after summing up the two potential functions does not use any edge of rank $E_{\geq i}$ then we know that we already found a rank-maximal matching. Otherwise we form G'_h by including only edges from E_i and continue the phases.

6 Conclusions

We presented a new algorithm which solves the rank-maximal matching problem by r maximum cardinality matching computations. Using the Hopcroft and Karp algorithm [16] we get time $O(min(n+r, r\sqrt{n})m)$ using linear space.

The algorithm is based on the idea of reducing the problem to the maximum weight matching problem. Doing this reduction implicitly, allows us to main-

tain the numbers appearing during the computation up to a reasonable level. Our algorithm answers an open question of [12], on whether such a running time can be achieved by using the maximum weight matching reduction.

What remains is to come up with a way of solving the remaining variants of the problem in the same running time.

References

[1] A. Abdulkadiroglu, T. Sonmez, Random serial dictatorship and the core from random endowments in house allocation problems, Econometrica 66 (3) (1998) 689–702.

[2] A. E. Roth, A. Postlewaite, Weak versus strong domination in a market with indivisible goods, Journal of Mathematical Economics 4 (1977) 131–137.

[3] L. Zhou, On a conjecture by Gale about one-sided matching problems, Journal of Economic Theory 52 (1) (1990) 123–135.

[4] A. Hylland, R. Zeckhauser, The efficient allocation of individuals to positions, Journal of Political Economy 87 (2) (1979) 293–314.

[5] D. J. Abraham, K. Cechlárová, D. Manlove, K. Mehlhorn, Pareto optimality in house allocation problems, in: ISAAC 2004: the 15th Annual International Symposium on Algorithms and Computation, 2005, pp. 1163–1175.

[6] D. J. Abraham, R. W. Irving, T. Kavitha, K. Mehlhorn, Popular matchings, in: SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005, pp. 424–432.

[7] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.

[8] H. N. Gabow, R. Tarjan, Faster scaling algorithms for network problems, SIAM Journal of Computing 18 (1989) 1013–1036.

[9] A. V. Goldberg, R. Kennedy, An efficient cost scaling algorithm for the assignment problem, Math. Program. 71 (2) (1995) 153–177.

[10] D. Abraham, N. Chen, V. Kumar, V. Mirrokni, Assignment problems in rental markets, in: Proceedings of WINE: Internet and Network Economics, Second International Workshop, Vol. 4286 of LNCS, 2006, pp. 198–213.

[11] T. Kavitha, personal communication (2007).

[12] R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, K. Paluch, Rank-maximal matchings, in: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2004, pp. 68–75.

[13] R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, K. Paluch, Rank-maximal matchings, ACM Transactions on Algorithms 2 (4) (2006) 1–9.

[14] M.-Y. Kao, T. W. Lam, W.-K. Sung, H.-F. Ting, A decomposition theorem for maximum weight bipartite matchings, SIAM Journal of Computing 31 (1) (2001) 18–26.

[15] K. Mehlhorn, S. Naher, LEDA: A Platform for Combinatorial and Geometric Computing, Cambridge University Press, 1999.

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