Bounded Unpopularity Matchings

Chien-Chung Huang[1] Telikepalli Kavitha[2] Dimitrios Michail[3] Meghana Nasre[2]

[1] Dartmouth College, USA. villars@cs.dartmouth.edu
[2] Indian Institute of Science, India. {kavitha,meghana}@csa.iisc.ernet.in
[3] INRIA Sophia-Antipolis, France. Dimitrios.Michail@sophia.inria.fr

Abstract. We investigate the following problem: given a set of jobs and a set of people with preferences over the jobs, what is the optimal way of matching people to jobs? Here we consider the notion of *popularity*. A matching M is popular if there is no matching M' such that more people prefer M' to M than the other way around. Determining whether a given instance admits a popular matching and, if so, finding one, was studied in [2]. If there is no popular matching, a reasonable substitute is a matching whose *unpopularity* is bounded. We consider two measures of unpopularity - *unpopularity factor* denoted by $u(M)$ and *unpopularity margin* denoted by $g(M)$. It has recently been shown that computing a matching M with the minimum value of $u(M)$ or $g(M)$ is NP-hard, and that if G does not admit a popular matching, then we have $u(M) \geq 2$ for all matchings M in G.

Here we show that a matching M that achieves $u(M) = 2$ can be computed in $O(m\sqrt{n})$ time (where m is the number of edges in G and n is the number of nodes) provided a certain graph H admits a matching that matches all nodes in $\mathcal{A}$. We also describe a sequence of graphs: $H = H_2, H_3, \ldots, H_k$ such that if H_k admits a matching that matches all nodes in $\mathcal{A}$, then we can compute in $O(km\sqrt{n})$ time a matching M such that $u(M) \leq k-1$ and $g(M) \leq n(1-\frac{2}{k})$. We ran our algorithm on random graphs and our simulation results suggest that our algorithm computes a matching with low unpopularity.

1 Introduction

The problem of assigning people to positions is a very common problem that arises in many domains. The input here is a bipartite graph $G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E})$, where nodes on one side of the bipartite graph rank edges incident on them in an order of preference, possibly involving ties. That is, the edge set $\mathcal{E}$ is partitioned into $\mathcal{E}_1 \dot\cup \mathcal{E}_2 \ldots \dot\cup \mathcal{E}_r$. We call $\mathcal{A}$ the set of *applicants*, $\mathcal{P}$ the set of *posts*, and $\mathcal{E}_i$ the set of edges with *rank i*. If $(a,p) \in \mathcal{E}_i$ and $(a,p') \in \mathcal{E}_j$ with $i < j$, we say that *a prefers p* to p'. If $i = j$, then a is *indifferent* between p and p'. The ordering of posts adjacent to a is called a's *preference list*. The problem is to assign applicants to posts that is *optimal* with respect to these preference lists.

This problem has been well-studied in economics literature, see for example [3, 13, 15]. It models some important real-world markets, including the allocation of graduates to training positions [7], families to government-owned housing [14], mail-based DVD rental systems such as NetFlix. Instances of these markets are can be regarded as restricted stable marriage instances [4, 6], in which members of one side of the market (posts) are indifferent between members of the other side of the market (applicants).

A *matching* M of G is a subset of E, such that no two edges of M share a common endpoint. Various criteria have been proposed to measure the "goodness" of a matching. For example, a matching is *Pareto-optimal* [1, 3, 13] if no applicant can improve his/her allocation (say by exchanging posts with another applicant) without requiring some other applicant to be worse off. There are many Pareto-optimal matchings and so we need stronger definitions: a matching is *rank-maximal* [8] if it allocates the maximum number of applicants to their first choice, and then subject to this, the maximum number to their second choice, and so on. Such a matching has the lexicographically maximum tuple $(n_1, n_2, \ldots)$ where n_i is the number of people assigned to positions they respectively rank i-th. A matching is *maximum utility* if it maximizes $\sum_{(a,p)\in M} u_{a,p}$, where $u_{a,p}$ is the utility of allocating post p to applicant a. Note that $u_{a,p}$ would be a function of the numerical rank that a associates with the edge (a,p). Thus most of these criteria use the actual values or numerical ranks expressed by applicants in their preference lists. Such criteria are easily prone to manipulation by people lying about their preferences. Moreover, the preference lists only express the "relative" ranking of the options. Measuring the optimality of a matching as a function of the actual numerical ranks may not be the correct approach. One criterion that does not use numerical ranks is *popularity*. We define it below.

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 $M(a)$ (where $M(a), M'(a)$ are the posts that a is matched to in M and in M', respectively).

Definition 1. *M' is* more popular than *M, denoted by $M' \succ M$, if the number of applicants that prefer M' to M is greater than the number of applicants preferring M to M'. A matching M is* popular *if there is no matching M' that is more popular than M.*

Figure 1 contains an example instance in which $\mathcal{A} = \{a_1, a_2, a_3\}$, $\mathcal{P} = \{p_1, p_2, p_3\}$, and each applicant prefers p_1 to p_2, and p_2 to p_3. Consider the three symmetrical matchings $M_1 = \{(a_1, p_1), (a_2, p_2), (a_3, p_3)\}$, $M_2 = \{(a_1, p_3), (a_2, p_1), (a_3, p_2)\}$ and $M_3 = \{(a_1, p_2), (a_2, p_3), (a_3, p_1)\}$. None of these matchings is popular, since $M_1 \prec M_2$, $M_2 \prec M_3$, and $M_3 \prec M_1$. In fact, it turns out that this instance admits no popular matching, the problem being that the *more popular than* relation is not transitive.

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 first polynomial-time algorithms for this problem were given in [2]: when there are no ties in the preference lists, the problem can be solved in $O(n+m)$ time, where $n = |\mathcal{A} \cup \mathcal{P}|$ and $m = |E|$, and more generally, the problem can be solved in $O(m\sqrt{n})$ time. The main drawback of the notion

$$\begin{array}{lcccc} a_1 & : & p_1 & p_2 & p_3 \\ a_2 & : & p_1 & p_2 & p_3 \\ a_3 & : & p_1 & p_2 & p_3 \end{array}$$

Fig. 1. An instance for which there is no popular matching.

of popular matchings is that such matchings may not exist in the given graph. In this situation, it would be desirable if we can find some good substitutes of a popular matching. This motivates our paper.

1.1 Problem Definition

In this paper, we assume that the input instance G does not admit a popular matching. Our goal is to compute a *least unpopular* matching. We use two criteria given by McCutchen [11] to measure the unpopularity of a matching. We first need the following definitions.

Given any two matchings X and Y in G, define $\phi(X,Y) =$ number of applicants that prefer X to Y. Let us define the following functions to compare two matchings X and Y:

$$\Delta(X,Y) = \begin{cases} \phi(Y,X)/\phi(X,Y) & \text{if } \phi(X,Y) > 0 \\ 1 & \text{if } \phi(X,Y) = 0 \text{ and } \phi(Y,X) = 0 \\ \infty & \text{otherwise.} \end{cases}$$

$$\text{and } \delta(X,Y) = \phi(Y,X) - \phi(X,Y).$$

Having the above functions, we can define the *unpopularity factor* of a matching M as:

$$u(M) = \max_{M'} \Delta(M,M').$$

The *unpopularity margin* of a matching M is defined as:

$$g(M) = \max_{M'} \delta(M,M').$$

The functions $u(\cdot)$ and $g(\cdot)$ were first introduced by McCutchen, who also gave polynomial time algorithms to compute $u(M)$ and $g(M)$ for any given matching M. A matching M is popular if and only if $u(M) = 1$ and $g(M) = 0$. When G does not admit popular matchings, we are interested in computing a matching M with a low value of $u(M)$. Suppose $u(M) \leq 2$. Then such a matching can be considered "reasonably popular" in a model where we say that a matching M' *beats* another matching M only when the number of applicants who prefer M' to M is more than twice the number of applicants who prefer M to M'. If $u(M) \leq 2$, then no other matching can beat M by the above rule. Note that all the 3 matchings M_1, M_2, M_3 described in Figure 1 have their u value equal to 2 and their g value equal to 1. Let us now define a *least unpopular* matching.

Definition 2. *A matching M which achieves the minimum value of $u(M)$ among all the matchings in G is defined as the least unpopularity factor matching in G. Similarly, a matching that achieves the minimum value of $g(M)$ among all matchings in G is defined as the least unpopularity margin matching in G.*

McCutchen recently showed that either computing a least unpopularity factor matching or a least unpopularity margin matching is NP-hard. He also showed that the unpopularity factor of any matching is always an integer. Thus when G does not admit a popular matching, the best matching in terms of the unpopularity factor that one can hope for in G is a matching M that satisfies $u(M) = 2$. Complementing McCutchen's results, we have the following new results here.

- A least unpopularity factor matching can be computed in $O(m\sqrt{n})$ time provided a certain graph H admits an $\mathcal{A}$-complete matching. Such a matching M that we compute in H satisfies $u(M) = 2$.
- We also show a more general result. We construct a sequence of graphs: $H = H_2, H_3, \ldots, H_k, \ldots$ and show that if H_k admits a matching that matches all nodes in $\mathcal{A}$, then we can compute in $O(km\sqrt{n})$ time a matching M such that $u(M) \leq k-1$ and $g(M) \leq n(1-\frac{2}{k})$.
- We ran our algorithm on random graphs using a similar setup as in [2]. Our simulation results suggest that when G is a random graph, then for values of $k \leq 4$, we see that H_k admits an $\mathcal{A}$-complete matching. Thus in these graphs our algorithm computes a matching M whose unpopularity factor is a number ≤ 3 and whose unpopularity margin can be upper bounded by $n/2$. We also give a probabilistic analysis to upperbound the performance of our algorithm.

1.2 Background and Related Results

The notion of popular matchings was first introduced by Gardenfors [5] in the context of the stable marriage problem. It is well known that every stable marriage instance admits a weakly stable matching (one for which there is no pair who strictly prefer each other to their partners in the matching). In fact, there can be an exponential number of weakly stable matchings, and so Gardenfors considered the problem of finding one with additional desirable properties, such as popularity. Gardenfors showed that when preference lists are strictly ordered, every stable matching is popular. He also showed that when preference lists contain ties, there may be no popular matching.

When only one side has preferences, Abraham et al. [2] gave polynomial time algorithms to find a popular matching, or to report none exists. Recently, Mahdian [9] showed that a popular matching exists with high probability, when (i) preference lists are randomly constructed, and (ii) the number of posts is a factor of $\alpha \approx 1.42$ larger than the number of applicants. He in fact showed a phase transition at α, that is, if the number of posts is smaller than α times the number of applicants, then with high probability popular matchings do not exist.

Manlove and Sng [10] generalized the algorithms of [2] to the case where each post has an associated *capacity*, the number of applicants that it can accommodate. (They described this in the equivalent context of the house allocation problem.) They gave an $O(\sqrt{C}n_1 + m)$ time algorithm for the no-ties case, and an $O((\sqrt{C}+n_1)m)$ time algorithm when ties are allowed, where n_1 is the number of applicants, m, as usual, is the total length of all preference lists, and C is the total capacity of all of the posts.

In [12] Mestre designed an efficient algorithm for the *weighted* popular matching problem, where each applicant is assigned a priority or weight, and the definition of popularity takes into account the priorities of the applicants. In this case the algorithm for the no-ties version has $O(n+m)$ complexity, and for the version that allows ties, the complexity is $O(\min(k\sqrt{n}, n)m)$, where k is the number of distinct weights assigned to applicants.

Organization of the paper. In Section 2 we describe the popular matching algorithm from [2], which is the starting point of our algorithm. We then describe McCutchen's algorithm to compute the unpopularity factor of a given matching. In Section 3 we describe our algorithm and bound its unpopularity factor and

unpopularity margin. In Section 4 we report our experimental results. Section 5 presents a probabilistic analysis of our algorithm.

2 Preliminaries

In this section, we review the algorithmic characterization of popular matchings given in [2] and the algorithm to compute the unpopularity index of a matching as given by McCutchen.

For exposition purposes, we create a unique strictly-least-preferred post $l(a)$ for each applicant a. In this way, we can assume that every applicant is matched, since any unmatched applicant a can be paired with $l(a)$. From now on, matchings are *applicant complete* (written as $\mathcal{A}$-complete). Also, without loss of generality, we assume that preference lists contain no gaps, i.e., if a is incident to an edge of rank i, then a is incident to an edge of rank $i-1$, for all $i > 1$.

Let $H_1 = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}_1)$ be the graph containing only rank-one edges. Then [2, Lemma 3.1] shows that a matching M is popular in G only if $M \cap \mathcal{E}_1$ is a maximum matching of H_1. Maximum matchings have the following important properties, which we use throughout the rest of the paper.

$M \cap \mathcal{E}_1$ defines a partition of $\mathcal{A} \cup \mathcal{P}$ into three disjoint sets: a node $u \in \mathcal{A} \cup \mathcal{P}$ is *even* (resp. *odd*) if there is an even (resp. odd) length alternating path in G_1 (w.r.t. $M \cap \mathcal{E}_1$) from an unmatched node to u. Similarly, a node u is *unreachable* if there is no alternating path from an unmatched node to u. Denote by $\mathcal{N}$, O and $\mathcal{U}$ the sets of even, odd, and unreachable nodes, respectively.

Lemma 1 (Gallai-Edmonds Decomposition). *Let $\mathcal{N}$, O and $\mathcal{U}$ be the sets of nodes defined by G_1 and $M \cap \mathcal{E}_1$ above. Then*

(a) $\mathcal{N}$, O and $\mathcal{U}$ are pairwise disjoint, and independent of the maximum matching $M \cap \mathcal{E}_1$.
(b) In any maximum matching of G_1, every node in O is matched with a node in $\mathcal{N}$, and every node in $\mathcal{U}$ is matched with another node in $\mathcal{U}$. The size of a maximum matching is $|O| + |\mathcal{U}|/2$.
(c) No maximum matching of G_1 contains an edge between a node in O and a node in $O \cup \mathcal{U}$. Also, G_1 contains no edge between a node in $\mathcal{N}$ and a node in $\mathcal{N} \cup \mathcal{U}$.

Using this node partition, we make the following definitions: for each applicant a, $f(a)$ is the set odd/unreachable posts amongst a's most-preferred posts. Also, $s(a)$ is the set of a's most-preferred posts amongst all even posts. We refer to posts in $\cup_{a \in \mathcal{A}} f(a)$ as *f-posts* and posts in $\cup_{a \in \mathcal{A}} s(a)$ as *s-posts*. Note that f-posts and s-posts are disjoint, and that $s(a) \neq \emptyset$ for any a, since $l(a)$ is always even. Also note that there may be posts in $\mathcal{P}$ that are neither f-posts nor s-posts. The next theorem characterizes the set of all popular matchings.

Theorem 1 ([2]). *A matching M is popular in G iff (i) $M \cap \mathcal{E}_1$ is a maximum matching of $H_1 = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}_1)$, and (ii) for each applicant a, $M(a) \in f(a) \cup s(a)$.*

Figure 2 contains the algorithm from [2], based on Theorem 1, for solving the popular matching problem.

2.1 McCutchen's algorithm

Here we outline the algorithm given by McCutchen for computing the unpopularity factor of a matching. Given a matching M, the idea is to find a series of promotions (of applicants) at the cost of demoting one applicant. The longest such promotion path determines the unpopularity factor of the particular matching. Such a path can be discovered by building a directed weighted graph on the set of posts. We will refer to

Popular-Matching$(G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}))$
Construct the graph $G' = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}')$, where $\mathcal{E}' = \{(a,p) : a \in \mathcal{A} \text{ and } p \in f(a) \cup s(a)\}$.
Construct a maximum matching M of $H_1 = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}_1)$.
//Note that M is also a matching in G'.
Remove any edge in G' between a node in O and a node in $O \cup \mathcal{U}$.
//No maximum matching of H_1 contains such an edge.
Augment M in G' until it is a maximum matching of G'.
Return M if it is $\mathcal{A}$-complete, otherwise return *"no popular matching"*.

Fig. 2. An $O(\sqrt{n}m)$-time algorithm for the popular matching problem (from [2]).

this graph as the Posts-Graph $G_{\mathcal{P}}$. The vertices of $G_{\mathcal{P}}$ represent all the posts $\mathcal{P}$ in the original graph. We add edges into $G_{\mathcal{P}}$ based on the following rules: (let $M(p)$ denote the applicant to which post p is matched to in the matching M)

- an edge with weight -1 is directed from post p_i to p_j if $M(p_i)$ prefers p_j to p_i.
- an edge with weight 0 is directed from post p_i to p_j if $M(p_i)$ is indifferent between p_i and p_j.

Note that there is no edge from p_i to p_j if $M(p_i)$ prefers p_i to p_j. The series of promotions mentioned above is a negative weight path in this graph. To find the longest negative weight path in this graph, we add a dummy vertex s with 0 weight edges from s to all posts. An algorithm which finds shortest paths from source s to all posts will give the longest negative weight path in $G_{\mathcal{P}}$. Existence of a negative weight cycle implies that there exists a promotion sequence without any demotion and hence the unpopularity factor of the matching is ∞. Let us assume that no negative weight cycles exist. Then all posts have a 0 or negative weight shortest path from the source. The post whose distance from the source is the most negative determines the unpopularity index of the matching M. For details of the proof of correctness, refer to [11].

3 Our algorithm

In this section we describe a *greedy* strategy to compute a matching of G, whose unpopularity can be bounded. Our algorithm is iterative and in every iteration it constructs a graph H_i and a maximum matching M_i in H_i. We show that if M_i is an $\mathcal{A}$-complete matching, then $u(M_i) \leq i-1$ and $g(M_i) \leq n(1-2/i)$.

We will first give some intuition before we formally describe our algorithm. Recall that the popular matching algorithm first finds a maximum cardinality matching M_1 in the graph H_1 (whose edge set is the set of all rank 1 edges). The algorithm then identifies all even applicants/posts using the Gallai-Edmonds decomposition and adds the edges (a,p) where a is even and $p \in s(a)$ to the pruned graph H_1 (all rank 1 edges between an odd node in H_1 and a node that is odd or unreachable in H_1 are removed from H_1). Note that each such edge (a,p) is *new* to H_1, that is, such an edge is not already present in H_1 since by Gallai-Edmonds decomposition (part (iii)), there is no edge between two *even* vertices of H_1, and here both a and p are even in H_1. In this new graph, call it H_2, M_1 is augmented to a maximum cardinality matching M_2. In case M_2 is $\mathcal{A}$-complete, we declare that the instance admits a popular matching. Otherwise no popular matching exists.

The idea of our algorithm here is an extension of the same strategy. Since we are considering instances which do not admit a popular matching, M_2 found above will not be $\mathcal{A}$-complete. In this case, we go further and find the Gallai-Edmonds decomposition of nodes in H_2 and identify nodes that are even in H_1 and in H_2. A node that is odd or unreachable in either H_1 or in H_2 will always be matched by a maximum cardinality matching in H_2 that is obtained by augmenting a maximum cardinality matching in H_1. Hence the nodes

that are not guaranteed to be matched by such a matching are the applicants and posts that are even in both H_1 and H_2.

So let us now add the edges (a,p) to H_2 where a and p are nodes that are even in both H_1 and H_2 and among all posts that are even in both H_1 and H_2, p is a most preferred post of a. We would again like to point out that such an edge (a,p) did not exist in either H_1 or in H_2, since a and p were even in H_1 and in H_2. We also prune H_2 to remove edges that are contained in no maximum cardinality matching of H_2 and call the resulting graph H_3. We then augment M_2 to get M_3 and continue the same procedure till we finally get an $\mathcal{A}$-complete matching M_i.

We would now like to contrast our approach above with the approach used in the algorithm for *rank-maximal* matchings [8]. In the i-th iteration the algorithm for rank-maximal matchings would add edges from an applicant a that is even in each of the previous iterations to a post p that was even in each of the previous iterations if and only if p was a rank i post in a's preference list. On the other hand, our algorithm will add an edge from an applicant a that is even in each of the previous iterations to a post q that is even in each of the previous iterations if q is a's most preferred post among all such posts. Note that the rank of the edge (a,q) is not necessarily i. Thus the absolute ranks in the preference lists are not important and instead, what is important here is the relative ordering of posts in each applicant's preference list. Thus unlike in the rank-maximal matching algorithm, in our algorithm every applicant a that has been even in all previous iterations will have some new edge incident on it in the i-th iteration.

With the above intuition, we are now ready to formally define the algorithm.

3.1 The algorithm

We start with $H_1 = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}_1)$ where $\mathcal{E}_1$ is the set of rank 1 edges. Let M_1 be any maximum cardinality matching in H_1.

Initialize $i = 1$ and let all nodes be unmarked.
While M_i is not $\mathcal{A}$-complete do:

1. Partition the nodes of $\mathcal{A} \cup \mathcal{P}$ into three disjoint sets: $\mathcal{N}_i, \mathcal{O}_i, \mathcal{U}_i$.
   – $\mathcal{N}_i$ and $\mathcal{O}_i$ consists of nodes that can be reached in H_i from an unmatched node by an even/odd length alternating path with respect to M_i, respectively.
   – $\mathcal{U}_i$ consists of nodes that are unreachable by an alternating path from any unmatched node in H_i.
2. Mark all unmarked nodes in $\mathcal{O}_i \cup \mathcal{U}_i$.
3. Delete all edges of H_i between a node in $\mathcal{O}_i$ and a node in $\mathcal{O}_i \cup \mathcal{U}_i$
4. Add edges (a,p) to H_i where (i) a in unmarked, (ii) p is unmarked and (iii) p is a's most preferred post among all unmarked posts. Call the resulting graph H_{i+1}.
5. Augment M_i in H_{i+1} to get a new matching M_{i+1} which is a maximum cardinality matching of H_{i+1}.
6. $i = i+1$.

Fig. 3. An $O(km\sqrt{n})$-time algorithm for finding an applicant-complete matching.

We note that once a post becomes odd or unreachable in any iteration, it gets marked and hence it cannot get new edges incident upon it in the subsequent iterations. We use this to show that the unpopularity factor of the matching that we produce is bounded by $k-1$ if we find an $\mathcal{A}$-complete matching in the graph H_k. The running time of our algorithm is determined by the least k such that H_k admits an $\mathcal{A}$-complete matching. Since each iteration of our algorithm takes $O(m\sqrt{n})$ time, the overall running time is $O(km\sqrt{n})$, where k is the least number such that H_k admits an $\mathcal{A}$-complete matching.

Before we prove our main theorems, we need the following definition that defines a *level j* post for an applicant a. A level 1 post for each applicant is just its rank 1 post. But from levels ≥ 2, a level j post for an applicant need not be its rank j post.

Definition 3. *A level j post for an applicant a is a post p such that (i) p is an even post in $H_1, \ldots, H_{j-1}$ and (ii) p is the most preferred post for a amongst all such posts.*

Theorem 2. *If our algorithm finds an $\mathcal{A}$-complete matching M_k in H_k, then $u(M_k) \leq k-1$.*

Proof. Let M_k be the $\mathcal{A}$-complete matching produced by our algorithm after k iterations. We draw the posts graph $G_{\mathcal{P}}$ corresponding to the matching M_k. The unpopularity index of M_k is the most negative distance of a vertex (post) in $G_{\mathcal{P}}$ from the dummy source s as described in Section 2. We now show that the posts in $G_{\mathcal{P}}$ can be partitioned into k layers (corresponding to the k iterations) such that all negative weight edges always go from higher numbered layers to lower numbered layers. If we show this, then it is clear that since there are only k layers and all negative weight edges have weight -1, the longest negative weight path can be of length at most $k-1$.

Let us partition the posts of $G_{\mathcal{P}}$ such that a post belongs to a layer t if it gets marked for the first time in iteration t. Let p be a post that belongs to level i. Recall that in $G_{\mathcal{P}}$ there is a negative weight edge from p to q iff $M_k(p)$ strictly prefers q to p. We now show that any such post q should belong to a layer j such that $j < i$.

First, note that an edge (a,p) is added to the graph at the end of the $(j-1)$-th iteration of our algorithm (for any $j \geq 1$) only if p is a level j post for a. Next, note that since p got marked in the i-th iteration, no new edges are ever added to p in any of the subsequent iterations. Based on these two observations, we can conclude that since the edge $(M_k(p), p)$ exists in $G_{\mathcal{P}}$ it has to be the case that p is a level ℓ post for $M_k(p)$ for some $\ell \leq i$.

That is, at the end of the $(\ell-1)$-th iteration, p was the most preferred *unmarked* post for $M_k(p)$. Hence all the posts that $M_k(p)$ strictly prefers to p were already marked before/during the $(\ell-1)$th iteration. That is, these posts belong to layers j, where $j \leq \ell - 1 \leq i-1$. Thus if (p,q) is a negative weight edge out of p, then q belongs to layer j, where $j < i$.

Hence we have shown that all negative weight edges must go from higher numbered layers to lower numbered layers. This implies that the longest negative weight path in the graph $G_{\mathcal{P}}$ corresponding to M_k is at most $k-1$. In other words, $u(M_k) \leq k-1$. □

Theorem 3. *If our algorithm finds an $\mathcal{A}$-complete matching M_k in H_k, then $g(M_k) \leq n(1-\frac{2}{k})$.*

Proof. Let M_k be the $\mathcal{A}$-omplete matching produced by our algorithm after k iterations and let M be any other $\mathcal{A}$-complete matching in G. Now let us construct a weighted directed graph $H_{\mathcal{P}}$ similar to the posts graph $G_{\mathcal{P}}$. The vertices of $H_{\mathcal{P}}$ are all posts p such that $M_k(p) \neq M(p)$. For every applicant a we have a directed edge from $M_k(a)$ to $M(a)$ with a weight of $-1, 0, +1$ if a prefers $M(a)$ better than, the same as or worse than $M_k(a)$. Any post p that does not belong to $H_{\mathcal{P}}$ is matched to the same applicant in M_k as well as in M and hence the corresponding applicant does not contribute to the unpopularity margin. Furthermore, it is clear that the sum of weights of all edges in $H_{\mathcal{P}}$ gives the negative of the unpopularity margin by which M dominates M_k.

First note that $H_{\mathcal{P}}$ is a set of disjoint paths and cycles. This is because, $H_{\mathcal{P}}$ can equivalently be constructed from $S = M_k \oplus M$ by striking off applicants and giving appropriate directions and weights to edges. Thus a path in S continues to be a path in $H_{\mathcal{P}}$ although it may no longer be of even length. The same is true for cycles also. If a path or cycle consists of only 0 weight edges, then we can drop such a cycle/path from the

graph, since these edges do not contribute to the unpopularity margin. In addition, note that any cycle or path cannot be composed of only negative and zero weight edges, otherwise the unpopularity factor of M_k is ∞, a contradiction. Hence we can assume that every cycle or path contains at least one positive edge.

Let ρ be any path or cycle in $H_{\mathcal{P}}$. Let us consider the function

$$\text{frac-margin}(\rho) = \frac{\text{number of } -1\text{'s in } \rho - \text{number of } +1\text{'s in } \rho}{\text{number of edges in } \rho}$$

Let us try to bound frac-margin(ρ) for each ρ. For the sake of simplicity, let us first assume that the preference lists are strict. So there are only ± 1 weight edges in $H_{\mathcal{P}}$. Thus frac-margin$(\rho) = (\alpha-\beta)/(\alpha+\beta)$ where α is the number of -1 weight edges in ρ and β is the number of $+1$ weight edges in ρ. Since the unpopularity factor of M_k is bounded by $k-1$, it is easy to see that the unpopularity factor of ρ is also bounded by $k-1$ (refer to [11] for a proof), thus $\alpha/\beta \leq k-1$. Thus $\beta/(\alpha+\beta) \geq 1/k$, and $\alpha/(\alpha+\beta) \leq 1-1/k$. Hence frac-margin($\rho$) for any path or cycle ρ is at most $1-2/k$. The contribution of ρ towards $\delta(M_k, M)$ is (number of edges in ρ)$\cdot$(frac-margin(ρ)). This is at most $n_\rho(1-2/k)$ where n_ρ is the number of edges in ρ. Since a unique applicant a is associated with each edge $(M_k(a), M(a))$ of $H_{\mathcal{P}}$, it follows that $\sum n_\rho \leq n$. Thus $\delta(M_k, M) \leq n(1-2/k)$ where M is any matching.

The proof for the case with ties also follows from the above argument. Since 0 weight edges in ρ do not affect the numerator of frac-margin(ρ) and only increase the denominator of frac-margin(ρ), it is easy to see that frac-margin(ρ) for a path or cycle ρ with 0 weight edges is dominated by frac-margin(ρ') where ρ' is obtained from ρ by contracting 0 weight edges. Thus frac-margin$(\rho) \leq 1-2/k$ and thus $\delta(M_k, M) \leq n(1-2/k)$ where M is any matching. Thus $max_{M'}\delta(M_k, M) \leq n(1-2/k)$. □

Corollary 1. *Let G be a graph that does not admit a popular matching. If our algorithm produces an $\mathcal{A}$-complete matching M in H_3, then M is a least unpopularity factor matching in G.*

Proof. It follows from Theorem 2 that if our algorithm produces an applicant complete matching M in H_3, then $u(M) \leq 2$. McCutchen [11] showed that the unpopularity factor of any matching is always an integer. Thus if G admits no popular matching, then the lowest value of $u(\cdot)$ we can hope for is 2. Since $u(M) \leq 2$, it follows that this is a least unpopularity factor matching. □

$a_1 : p_1 p_8 \cdots$
$a_2 : p_2 p_9 \cdots$
$a_3 : p_3 p_{10} \cdots$
$a_4 : p_4 p_1 \cdots$
$a_5 : p_4 p_1 p_5 p_6 p_7 \cdots$
$a_6 : p_4 p_2 p_5 p_6 p_7 \cdots$
$a_7 : p_4 p_3 p_5 p_6 p_7 \cdots$

Fig. 4. All preference lists are strictly-ordered.

Starting with the above corollary, it is tempting to push the frontier further. Suppose that the algorithm gets an $\mathcal{A}$-complete matching M_4 in the graph H_4 (thus $u(M_4) \leq 3$). Can we also argue that it is impossible to achieve a better matching? Unfortunately, this is not the case. In the example given in Figure 4, we show a problem instance, where our algorithm terminates in constructing $M_4 = \{(a_1, p_1), (a_2, p_2), (a_3, p_3), (a_4, p_4), (a_5, p_5), (a_6, p_6), (a_7, p_7)\}$ in H_4. However, the matching $M^* = \{(a_1, p_8), (a_2, p_9), (a_3, p_{10}), (a_5, p_1), (a_6, p_2),$

$n = 100$				
		# rounds		
k	t	2	3	4
10	0.05	4	996	
	0.2	28	972	
	0.5	471	529	
	0.8	729	271	
	1.0	1000		
25	0.05		991	9
	0.2	3	991	6
	0.5	138	861	1
	0.8	773	227	
	1.0	1000		
50	0.05		948	52
	0.2	1	978	21
	0.5	158	832	10
	0.8	793	207	
	1.0	1000		
100	0.05		952	48
	0.2	2	973	25
	0.5	148	836	16
	0.8	783	217	
	1.0	1000		

$n = 500$				
		# rounds		
k	t	2	3	4
10	0.05		1000	
	0.2		1000	
	0.5	176	824	
	0.8	62	938	
	1.0	1000		
25	0.05		1000	
	0.2		1000	
	0.5		999	1
	0.8	93	907	
	1.0	1000		
50	0.05		967	33
	0.2		994	6
	0.5		997	3
	0.8	104	896	
	1.0	1000		
100	0.05		828	172
	0.2		942	58
	0.5		989	11
	0.8	93	907	
	1.0	1000		

n	# rounds		
	2	3	4
10	585	413	2
25	141	844	15
50	6	962	32
100		952	48
250		896	104
500		820	180
1000		667	333
1500		541	459
2000		320	680

Table 1. The left and middle tables show the number of instances with $n = 100$ and 500 nodes respectively (out of a 1000 instances) that finish in round number 2 (popular matching), 3 or 4 for different values of the parameters k and t. The table on the right shows the number of instances (out of a 1000 instances) that finish in round number 2 (popular matching), 3 or 4 for fixed $t = 0.05$, $k = n$ and different values of the parameter n.

$(a_7, p_3), (a_4, p_4)\}$ does achieve $u(M^*) = 2$.

It is easy to extend the above instance by adding applicants and posts so that the greedy algorithm takes as large number of iterations as desired even when the instance admits a matching with $u(M) = 2$.

4 Experimental Results

Previous work [2] regarding popular matchings presented simulation results about the existence of popular matchings in random graphs. A follow-up work studied theoretically the necessary conditions for a random graph to admit a popular matching [9]. On the other hand the algorithm that we propose in this paper is guarantied to return some matching even if the instance does not admit a popular one. In this section we study the number of rounds that it requires until it returns a solution. Although the unpopularity grows linearly with the number of rounds, our simulation results suggest that in random graphs the necessary rounds are a small constant.

We follow the setting used in [2] in order for our experimental results to be comparable to those reported in [2]. The number of applicants and posts are equal (denoted by n) and preference lists have the same length k. Existence of ties is characterized by a single parameter t which denotes the probability of an entry in the preference list to be tied with its predecessor.

Table 1 contains simulation results for random graphs with $n = 100$ and $n = 500$ for different values of parameters k and t. The table shows the number of instances (out of 1000 instances) that finish in some particular round of the execution. Round 2 means that the instance has a popular matching. It is easy to

observe that the difficult cases are the ones where we only have a few ties (t is small). For a fixed value of k as t decreases the algorithm requires more rounds until it returns a solution. However, the good news are that it never takes more than four rounds.

As Table 1 suggests the difficult situation is when k is large (roughly n) and t is very small (the preferences have few ties and these ties are of small length). We study this situation further by varying the value of n in order to see whether our observations for $n = 100$ and $n = 500$ are valid for larger values of n. Let us remark that Mahdian [9] proved the following result. If the right side of the bipartite partition is slightly larger than (≈ 1.42 times) the left side, then the instance has a popular matching with high probability. Since we try to identify difficult instances we keep the two sides of the partition equal, which is also the case in many practical situations, where there are no surplus projects/posts when compared to the number of applicants.

Table 1 shows the number of rounds (again out of a 1000) that is required for different values of n when $t = 0.05$ and $k = n$, i.e., the graph is complete bipartite and there only a few ties of very small length exist. The table suggests that as n increases the probability of terminating at the second or third round decreases while the one of the 4th round increases. However, this is not accompanied with any increase in rounds larger than 4. Due to memory constraints we could not continue the experiment for larger values of n, but we conjecture that for larger values of n all instances will terminate in round 4.

Our experimental results are very promising. The algorithm behaves nicely in practice, far away from a possible large approximation.

5 A bound on the number of iterations taken by our algorithm

In this section we probabilistically bound the number of iterations our algorithm takes to compute an $\mathcal{A}$-complete matching. We show that, on random instances, the expected number of iterations taken by our algorithm is at most $\lceil \ln n \rceil$. In this section, we assume that each preference list is complete and has no ties in it. Each preference list is a uniform random permutation on the set of all posts. We also assume that the number of applicants is equal to the number of posts, and let us call this number n.

We now describe our random experiment. Each applicant picks a permutation independently and uniformly at random from the set of all permutations on the posts. For the sake of analysis of our algorithm, we view the experiment in a slightly different manner as was done in [9]. Each applicant picks his/her first choice post independently and uniformly at random from the set of posts $\mathcal{P}$. We denote the set of first posts thus picked by all the applicants as L_1. If a post in L_1 is sought by more than 1 applicant, then we arbitrarily assign this post to one of the applicants seeking it. So the applicants that do not have a post assigned to them have to try again in the next round whereas the $|L_1|$ applicants who have found posts do not pick any post in further rounds. So at the end of round 1, we are left with $n - |L_1|$ unmatched posts and $n - |L_1|$ unmatched applicants. It is easy to see that the expected value $E[n - |L_1|]$ is

$$E[n - |L_1|] = n(1 - \frac{1}{n})^n \leq \frac{n}{e}.$$

Each of these $n - |L_1|$ unmatched applicants further picks his/her level 2 post independently and uniformly at random from $\mathcal{P} - L_1$. This experiment is identical to the experiment in the first round, except that we are operating with $n - |L_1|$ applicants and $n - |L_1|$ posts instead of n applicants and n posts. In round i, we will have n_i applicants and n_i posts and each of these applicants picks his/her level i post independently and uniformly at random from $\mathcal{P} - \{L_1 \cup L_2 \cup \cdots \cup L_{i-1}\}$. It is easy to see that the expected value of n_i is $n(1 - 1/n)^{n(i-1)} \leq n/e^{i-1}$. Thus at the end of round $\lceil \ln n \rceil$ we expect to have no empty posts, that is, we have a perfect matching.

Note that the number of rounds taken by the above experiment is an upper bound on the number of rounds taken by our algorithm, since for the simplicity of analysis, in the above experiment in each round we assigned a post that is sought by more than one applicant to arbitrarily one of them, whereas in our algorithm we make no such arbitrary assignment. Thus the number of rounds taken by our algorithm to find a perfect matching is possibly much lower than the number of rounds taken by the above experiment to find a perfect matching, as the experimental results attested. Note that we can also show that the number of rounds in the above experiment is with high probability $c \ln n$, for a small constant c, using Azuma's Inequality.

References

1. D.J. Abraham, K. Cechlárová, D.F. Manlove, K. Mehlhorn. *Pareto-optimality in house allocation problems*. In *Proceedings of ISAAC 2004: the 15th Annual International Symposium on Algorithms and Computation*, (LNCS 3341), pages 3–15, 2004.
2. D.J. Abraham, R.W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. SIAM Journal on Computing, Vol.37, No.4, pp. 1030–1045, 2007. (Preliminary version in *Proc. of 16th SODA*, pages 424-432, 2005.)
3. A. Abdulkadiroğlu and T. Sönmez. *Random serial dictatorship and the core from random endowments in house allocation problems*. Econometrica, 66(3):689–701, 1998.
4. D. Gale and L.S. Shapley. *College admissions and the stability of marriage*. American Mathematical Monthly, 69:9–15, 1962.
5. P. Gardenfors. *Match making: assignments based on bilateral preferences*. Behavioural Sciences, 20:166–173, 1975.
6. D. Gusfield and R.W. Irving *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.
7. A. Hylland and R. Zeckhauser. *The efficient allocation of individuals to positions*. Journal of Political Economy, 87(2):293–314, 1979.
8. R.W. Irving and T. Kavitha and K. Mehlhorn and D. Michail and K. Paluch. *Rank-maximal matchings.* In *Proceedings of the 15th annual ACM-SIAM symposium on Discrete algorithms*, pages 68–75, 2004.
9. M. Mahdian. Random popular matchings. In *Proceedings of the 7th ACM Conference on Electronic-Commerce*, pages 238–242, 2006.
10. D.F. Manlove and C. Sng. Popular matchings in the capacitated house allocation problem. In *Proceedings of ESA 2006, the 14th Annual European Symposium on Algorithms*, (LNCS 4168), pages 492-503, Springer-Verlag, 2006.
11. M. McCutchen. Least-Unpopularity-Factor Matching. Manuscript, 2006.
12. J. Mestre. *Weighted popular matchings.* In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, (LNCS 4051), pages 715–726, 2006.
13. A.E. Roth and A. Postlewaite. *Weak versus strong domination in a market with indivisible goods*. Journal of Mathematical Economics, 4:131–137, 1977.
14. Y. Yuan. *Residence exchange wanted: a stable residence exchange problem*. European Journal of Operational Research, 90:536–546, 1996.
15. L. Zhou. *On a conjecture by Gale about one-sided matching problems*. Journal of Economic Theory, 52(1):123–135, 1990.