Stable Matchings with One-Sided Ties and Approximate Popularity

Telikepalli Kavitha
Tata Institute of Fundamental Research, Mumbai, India

Abstract

We consider a matching problem in a bipartite graph $G = (A \cup B, E)$ where vertices in A rank their neighbors in a strict order of preference while vertices in B are allowed to have *weak* rankings, i.e., ties are allowed in their preferences. Stable matchings always exist in G and are easy to find, however popular matchings need not exist and it is NP-complete to decide if one exists. This motivates the "approximately popular" matching problem.

A well-known measure of approximate popularity is *low unpopularity factor*. We show that when each tie in G has length at most k, there always exists a stable matching whose unpopularity factor is at most k. Our proof is algorithmic and we compute such a stable matching in polynomial time. Our result can be considered to be a generalization of Gärdenfors' result (1975) which showed that when rankings are strict, every stable matching is popular.

There are several applications where the size of the matching is its most important attribute. What one seeks here is a maximum matching M such that there is no maximum matching more popular than M. When rankings are weak, it is NP-hard to decide if G admits such a matching. When ties are one-sided and of length at most k, we show a polynomial time algorithm to find a maximum matching whose unpopularity factor *within* the set of maximum matchings is at most $2k$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Bipartite graphs, Maximum matchings, Unpopularity factor

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2022.22

Funding *Telikepalli Kavitha*: Supported by the Department of Atomic Energy, Government of India, under project no. RTI4001.

Acknowledgements Thanks to the reviewers for their helpful comments and suggestions.

1 Introduction

Our input is a bipartite graph $G = (A \cup B, E)$ where vertices in A are called *agents* and those in B are called *jobs*. Every vertex ranks its neighbors in an order of preference – while every agent ranks its neighbors in a strict order of preference, jobs have weak rankings, i.e., ties are allowed in their preferences. The above model is well-studied and such a model is seen when matching applicants to jobs or students to projects, e.g., the Scottish Foundation Allocation Scheme (SFAS) [21].

So in our model, every agent has a strict ordering of jobs that she finds interesting, however every job need not come up with a total order on all interested agents – here agents get grouped together in terms of their suitability to do this job, thus equally competent agents are tied together at the same rank. Our goal is to find an optimal matching in G. The classical notion of optimality is *stability*.

A matching M is stable if there is no edge that *blocks* M; an edge (a, b) is said to block M if both a and b prefer each other to their respective assignments[1] in M. The Gale-Shapley algorithm [10] (where ties are broken arbitrarily) finds a stable matching in G in linear time.

[1] Note that being left unmatched is the least preferred option for any vertex.


© Telikepalli Kavitha;
licensed under Creative Commons License CC-BY 4.0

Popularity. Another natural notion of optimality is *popularity*. Given any pair of matchings M and N, we say a vertex v prefers M to N if v prefers its assignment in M to its assignment in N. Let $\phi(M, N)$ be the number of vertices that prefer M to N and similarly, let $\phi(N, M)$ be the number of vertices that prefer N to M. Matching N is *more popular* than matching M if $\phi(N, M) > \phi(M, N)$.

▶ **Definition 1.** *A matching M is popular if there is no matching more popular than M, i.e., $\phi(M, N) \geq \phi(N, M)$ for all matchings N.*

Both stability and popularity are very desirable properties for a matching in G. While stability captures the important property that there is no pair (a, b) where both a and b are better-off by deviating from their respective assignments and pairing up with each other, popularity lays emphasis on aggregate or majority. So popularity ensures that a majority vote cannot force a migration to another matching.

When preferences are strict, stability is a stronger property and every stable matching is popular[2] [11]. So popular matchings always exist in G and the Gale-Shapley algorithm finds one. This is our ideal situation as we have a matching that is stable and hence, popular.

However when ties are allowed in preferences, the situation is no longer ideal – stability does not imply popularity. Even worse, popular matchings need not exist and it is NP-hard to decide if one exists [3]. Moreover, this hardness holds even in the setting of one-sided ties and every tie has length at most three [6]. Thus the popular matching problem with one-sided ties is NP-hard even under such a strict restriction on tie lengths.

Relaxing popularity. When ties are present in preferences, the lack of existence of popular matchings (and the hardness of solving the popular matching problem) motivates relaxing popularity to *near-popularity* or "low unpopularity". A well-studied measure of unpopularity of a matching is *unpopularity factor*, introduced in [22] and studied, e.g., in [2, 8, 13, 17, 24]. Given a matching M, its unpopularity factor $u(M)$ is defined below. For any matching N,

$$\text{let } \lambda(M, N) = \begin{cases} \phi(N, M)/\phi(M, N) & \text{if } \phi(M, N) > 0; \\ 1 & \text{if } \phi(M, N) = \phi(N, M) = 0; \\ \infty & \text{otherwise.} \end{cases}$$

Define $u(M) = \max_N \lambda(M, N)$. Thus in an election involving M and any matching N, the number of vertices that prefer N is at most $u(M)$ times the number that prefer M.

A matching M is popular if and only if $u(M) = 1$. A matching M with a low value of $u(M)$ is considered to be *near-popular*. Such a matching may lose elections but there are no heavy losses. Do near-popular matchings always exist in G? Consider an instance $K_{n,n}$ where every agent has the same preference order $b_1 \succ b_2 \succ \cdots \succ b_n$ (here $b_1, \ldots, b_n$ are the n jobs) while every job b_i has a tie of length n, i.e., it is indifferent between any two agents. It is easy to check that any matching in this instance has unpopularity factor at least $n - 1$. Thus there is *no* near-popular matching here.

However jobs in this instance have ties of length n in their preferences. Suppose no tie is very long, say every tie has length at most k (for some appropriate k). Such a restriction is quite natural in real-world applications; in fact, in many instances, we would have $k = O(1)$. Do near-popular matchings always exist then? Furthermore, are there such stable matchings?

[2] In an election between a stable matching S and any matching M, if vertex v prefers M to S then $M(v)$ has to prefer S to M, otherwise $(v, M(v))$ blocks S, which is forbidden. Hence $\phi(M, S) \leq \phi(S, M)$.

When preferences involve ties (even for one-sided ties of length two), a stable matching may have an unbounded unpopularity factor. Consider the following instance where $A = \{a_1, \ldots, a_n\}, B = \{b_1, \ldots, b_n\}$, and the preferences of vertices are as follows:

$$a_1 : \quad b_1 \succ b_n \qquad\qquad a_i : \quad b_{i-1} \succ b_i \quad \text{for } 2 \le i \le n$$
$$b_n : \quad a_1 \sim a_n \qquad\qquad b_i : \quad a_i \sim a_{i+1} \quad \text{for } 1 \le i \le n-1.$$

So a_1's top choice is b_1 and second choice is b_n and for $i \ge 2$, a_i's top choice is b_{i-1} and second choice is b_i. Every $b_i \in B$ has exactly two neighbors and it is indifferent between them. The matching $M = \{(a_i, b_i) : 1 \le i \le n\}$ is stable. Let $N = \{(a_1, b_n)\} \cup_{i=2}^{n} \{(a_i, b_{i-1})\}$. Observe that $\phi(N, M) = n - 1$ and $\phi(M, N) = 1$, so $u(M) \ge n - 1$. It is easy to check that N is a popular matching and it is also stable.

Is there always a matching (even better, a stable matching) with low unpopularity factor in an instance with one-sided ties of bounded length? We show the following result here.

▶ **Theorem 2.** *Let $G = (A \cup B, E)$ be an instance with one-sided ties of length at most k. There is always a stable matching M in G such that $u(M) \le k$. Furthermore, M can be computed in $O(k^2 \cdot mn)$ time, where $|A| = n$ and $|E| = m$.*

Observe that our earlier example of the instance $K_{n,n}$ (where each job has a tie of length n in its preference list) shows that Theorem 2 is almost tight. Recall that every matching in this instance $K_{n,n}$ has unpopularity factor at least $n - 1$. Thus when jobs have ties of length k, there are instances where every matching has unpopularity factor at least $k - 1$. Interestingly, we are able to show that any such instance always admits a *stable* matching with unpopularity factor at most k.

The above result generalizes the result of Gärdenfors [11] that a popular matching always exists in a bipartite graph G when rankings are strict (so $k = 1$ here). Indeed, when rankings are strict, every stable matching is popular. When ties are one-sided and each tie has length at most k, though every stable matching need not have a bounded unpopularity factor, our algorithm shows that there is always at least one stable matching M with $u(M) \le k$. For small values of k, the *near-popularity* of M can be justified as follows: in any election involving M, if the votes in favor of M are scaled by a factor of k and we sum up these weighted votes then M never loses any election.

Maximum matchings and near-popularity. There are several applications, e.g., matching medical students to residencies [4] or allocation problems in humanitarian organizations [25, 26], where the size of the matching is more important than stability or popularity. What one seeks here is a *best* maximum matching. Let us define a "best maximum matching" as one that does not lose to another maximum matching. When rankings are strict, it is known that such a maximum matching always exists and there is an $O(mn)$ time algorithm to find one such matching [17], where $|A| = n$ and $|E| = m$.

When vertices on one side are allowed to have weak rankings, it is NP-hard to decide if such a matching exists [6]. The hardness proof in [6] showed that it is NP-hard to find a popular matching when vertices of B are allowed to have weak rankings. It is easy to check that the same proof also shows it is NP-hard to find in such an instance a maximum matching that does not lose to any maximum matching. We show the following result here.

▶ **Theorem 3.** *Let $G = (A \cup B, E)$ be an instance with one-sided ties of length at most k. There always exists a maximum matching M in G such that $\phi(N, M) \le 2k \cdot \phi(M, N)$ for any maximum matching N; furthermore, M can be computed in $O(k^2 \cdot mn^2)$ time.*

Thus we can find in polynomial time a maximum matching whose unpopularity factor *within* the set of maximum matchings is at most $2k$.

1.1 Background and related results

In the setting of strict preferences or strict rankings, popularity is a well-studied relaxation of stability – see [5] for a survey. In the setting of ties in preferences, as mentioned earlier, stability does not imply popularity; moreover, it is NP-hard to decide if a popular matching exists [3, 6], thus it is NP-hard to find a least unpopularity factor matching. Another problem that is easy for strict preferences but NP-hard for ties (even for one-sided ties) is the maximum stable matching problem which asks for a stable matching of largest size. This problem is very well-studied and several approximation algorithms in the setting of one-sided ties are known [1, 14, 15, 16, 18, 19].

Near-popularity. To the best of our knowledge, no positive results on near-popular matchings in the setting of one-sided ties are currently known. Unpopularity factor $u(\cdot)$ is a well-studied measure of unpopularity of a matching. A size-unpopularity factor trade-off in an instance $G = (A \cup B, E)$ with strict rankings is known: for any integer $k \geq 2$, there exists a matching M_k such that $u(M_k) \leq k-1$ and $|M_k| \geq \frac{k}{k+1} \cdot |M^*|$, where M^* is a maximum matching, and such a matching M_k can be efficiently computed [17]. As shown in [17], this trade-off leads to the result that there always exists a maximum matching that is popular within the set of maximum matchings and it can be efficiently computed. In contrast to this, finding a maximum matching with the minimum number of blocking edges is NP-hard and this is NP-hard to approximate within $n^{1-\varepsilon}$, for any $\varepsilon > 0$ [4].

Matchings with low unpopularity factor in dynamic matching markets were studied in [2]. In the setting of bipartite graphs with strict rankings, when there are edge costs, it is NP-hard to find a min-cost popular matching [9]. Polynomial-time algorithms were given in [8] to find a *quasi-popular* matching M, i.e., $u(M) \leq 2$, whose cost is at most that of a min-cost popular matching. Popular matchings have also been studied in non-bipartite graphs with strict rankings. Popular matchings need not exist in such an instance G and it is NP-complete to decide if there exists one [9, 12]. It was shown in [13] that G always admits a matching with unpopularity factor $O(\log n)$ and there are non-bipartite instances with strict rankings where every matching has unpopularity factor $\Omega(\log n)$.

1.2 Our techniques

Our algorithm, roughly speaking, constructs a subgraph G' of G and returns a maximum matching in G'. The construction of G' is via a subroutine called Propose$(\cdot)$, where the argument is an agent a left unmatched in at least one maximum matching in the current G'. Such a vertex a is called *even* (formally defined in Section 2). Similar to the Gale-Shapley algorithm, in the subroutine Propose(a), the agent a proposes to its neighbors in G one-by-one in decreasing order of preference.

Any vertex in B always prefers proposals from better-ranked neighbors to worse-ranked neighbors. Recall that ties are allowed in the preferences of vertices in B. So how does a job b choose between proposals of two neighbors a and a' that are tied in its ranking? The following simple rule will be key to bounding the unpopularity factor of our matching:

- Among neighbors tied in its ranking, b prefers proposals from neighbors that place b high in their preference order of neighbors in G'.

To answer our above question on a versus a' when they are tied in b's ranking, if a (resp., a') regards b as its i-th-ranked (resp., j-th ranked) neighbor in G', then b prefers a's proposal if $i < j$, it prefers a''s proposal if $j < i$, else (so $i = j$) the proposals are tied.

While $a \in A$ is even in G' and its degree is less than k, it is allowed to propose in Propose(a) to its neighbors in G. When no vertex in A is even in G', we compute a maximum matching M in the final subgraph G' and show that M is a stable matching in G with $u(M) \leq k$. This algorithm and its analysis are given in Section 3.

In order to find a desired maximum matching (as given in Theorem 3), we run the above algorithm from Section 3 in a new graph H. The graph H is a multigraph where every edge in G is replicated n times. The construction of H is inspired by the popular maximum matching algorithm from [17] where every $a \in A$ gets n chances to propose to its neighbors. Bounding the unpopularity factor of the matching computed by this algorithm within the set of maximum matchings is trickier than the analysis given in Section 3. This algorithm and its analysis are given in Section 4.

2 Preliminaries

Our algorithm will use the classical *Dulmage-Mendelsohn* decomposition [7]. Let $G = (A \cup B, E)$ be a bipartite graph and let M be a matching in G. An alternating path with respect to M is a path whose alternate edges are in M. We have $A \cup B = \mathcal{E}_M \cup \mathcal{O}_M \cup \mathcal{U}_M$, where a vertex v is in $\mathcal{E}_M$ (resp., $\mathcal{O}_M$) if there is an even (resp., odd) length alternating path with respect to M from a vertex left unmatched in M to v and a vertex v is in $\mathcal{U}_M$ if there is no alternating path from an unmatched vertex to v.

The sets $\mathcal{E}_M$, $\mathcal{O}_M$, and $\mathcal{U}_M$ will be called the sets of *even, odd*, and *unreachable* vertices, respectively, with respect to M. The following theorem is well-known [20, 23].

▶ **Theorem 4.** *The sets $\mathcal{E}_M$, $\mathcal{O}_M$, and $\mathcal{U}_M$ are pairwise disjoint if and only if M is a maximum matching in G. Any maximum matching in G partitions the vertex set into the same sets of even, odd, and unreachable vertices. Furthermore,*

1. *There is no edge between the sets $\mathcal{E}_M$ and $\mathcal{E}_M \cup \mathcal{U}_M$, i.e., all the neighbors of vertices in $\mathcal{E}_M$ are in $\mathcal{O}_M$.*
2. *For any maximum matching N, we have $N \subseteq (\mathcal{O}_M \times \mathcal{E}_M) \cup (\mathcal{U}_M \times \mathcal{U}_M)$. Also $|N| = |\mathcal{O}_M| + |\mathcal{U}_M|/2$, hence all vertices in $\mathcal{O}_M \cup \mathcal{U}_M$ are matched in N.*

3 Our algorithm for a near-popular matching

We present an iterative algorithm to compute a desired stable matching in an instance $G = (A \cup B, E)$ with one-sided ties, where each tie has length at most k. A relevant graph for us is $G_0 = (A \cup B_0, E_0)$ where $B_0 = B \cup \{d(a) : a \in A\}$, i.e., for each $a \in A$, we add a corresponding dummy vertex $d(a)$ called a's *last resort job*, and $E_0 = E \cup \{(a, d(a)) : a \in A\}$. So $d(a)$ has only a as its neighbor and $d(a)$ will be the worst ranked neighbor of a.

Algorithm 1 is the overall algorithm. This algorithm constructs a subgraph G' of G_0. The while-loop here calls the subroutine Propose until no agent is *even* (see Section 2) in the current G'. Recall that vertices in A or agents have strict rankings. In our algorithm, any agent a such that – (i) a is *even* in the current G' and (ii) a's degree is less than k in the current G' – will be allowed to propose, i.e., a will be allowed to add some edges to the current G' in the subroutine Propose(a).

We will show in Lemma 6 that if the set of even agents in G' is non-empty then there has to be an even agent with degree less than k in G'. Such an agent a will propose in the subroutine Propose(a). When no agent is even in the subgraph G', the construction of G' is complete. Any maximum matching in the final G' will be returned by Algorithm 1.

Algorithm 1 Finding a stable matching M with $u(M) \leq k$ in $G = (A \cup B, E)$.

1: Initialize $G_0 = (A \cup B_0, E_0)$ and $G' = (A \cup B_0, \emptyset)$.
2: **while** there exists an *even* agent in G' **do**
3: Let a be any even agent in G' such that $\deg_{G'}(a) < k$.
4: Call Propose(a).
5: Return a maximum matching M in G'.

In the subroutine Propose(a) (see Algorithm 2), a proposes to its neighbors in G_0 one-by-one in decreasing order of preference. Each time the subroutine Propose(a) is called, all the proposals of a made in previous invocations of Propose(a) are forgotten (see line 1 of Algorithm 2) and a proposes afresh starting from its most favorite neighbor in G_0.

Algorithm 2 The subroutine Propose(a).

1: Delete from G' all edges incident to a. ▷ (*so all the previous proposals of a are forgotten*)
2: Set $i = 1$.
3: **repeat**
4: Let $b = a$'s most favorite neighbor in G_0 that a has not (freshly) proposed to.
5: Set $\mathsf{rank}_{G'}(a, b) = i$.
6: Delete from G_0 and G' all edges between b and b's neighbors ranked worse than a.
▷ (*so b will keep in G' only the best proposals that it receives in the entire algorithm*)
7: **if** b is isolated in G' **then**
8: Add the edge (a, b) to G'.
9: **else** ▷ (*a is tied with b's neighbors in G'*)
10: Let a' be any neighbor of b in G'.
11: **if** $i < \mathsf{rank}_{G'}(a', b)$ **then** ▷ (*so* $\mathsf{rank}_{G'}(a, b) < \mathsf{rank}_{G'}(a', b)$)
12: Delete all edges incident to b in G'.
13: Add the edge (a, b) to G'.
14: **if** $i = \mathsf{rank}_{G'}(a', b)$ **then** ▷ (*so* $\mathsf{rank}_{G'}(a, b) = \mathsf{rank}_{G'}(a', b)$)
15: Add the edge (a, b) to G' and set $i = i + 1$.
16: **until** $i = k + 1$ or a is no longer even in G'.

When $b \in B$ receives a proposal from a in Propose(a), the edge (a, b) will get added to the graph G' if one of the following two conditions is satisfied:

1. b prefers a to its current neighbors in G' – then (a, b) is unconditionally added to G'.
2. a is tied with b's neighbors in G' and a certain value called $\mathsf{rank}_{G'}(a, b)$ is good enough.

When a proposes to b, $\mathsf{rank}_{G'}(a, b)$ will be b's rank among a's neighbors in the current G'. Let $\mathsf{rank}_{G'}(a, b) = i$. Suppose a and a' are tied in b's ranking, where a' is a neighbor of b in G'. When a proposes to b, i.e., when we consider the edge (a, b) in Propose(a):

- if $i < \mathsf{rank}_{G'}(a', b)$ then b deletes all edges incident to it in G' and the edge (a, b) is added to G' (see line 13).
- if $i = \mathsf{rank}_{G'}(a', b)$ then b retains all its current edges in G' and the edge (a, b) is also added to G' (see line 15). When a proposes to its next most favorite neighbor (say, b') in G_0, it will be the case that $\mathsf{rank}_{G'}(a, b') = i + 1$.
- if $i > \mathsf{rank}_{G'}(a', b)$ then the edge (a, b) is not added to G'. When a proposes to its next most favorite neighbor (say, b') in G_0, it will be the case that $\mathsf{rank}_{G'}(a, b') = i$.

Thus among the best proposals (wrt b's ranking) that b receives in the entire algorithm, b keeps edges in G' to only those neighbors a such that $\mathsf{rank}_{G'}(a,b)$ is the *minimum*. In $\mathsf{Propose}(a)$ when the edge (a,b) gets added to G' in line 8 or line 13, a will be the only neighbor of b in G'. This makes a *unreachable*[3] in G' – so a is no longer *even* in G' and this will be the last iteration of the repeat-loop in this invocation of $\mathsf{Propose}(a)$.

If a is even in G', then there is one more iteration if $i \leq k$. Observe that i is increased when $\deg_{G'}(a)$ gets increased in line 15. Thus we will always ensure that $\deg_{G'}(a) \leq k$. Let us recall the other useful invariant that is maintained in our algorithm:

- all neighbors of a job b in G' are tied in b's ranking.

▶ Remark 5. After the edge (a,b) is added to G', the graph G' may change in later invocations of $\mathsf{Propose}(\cdot)$. But $\mathsf{rank}_{G'}(a,b)$ remains the same till the next invocation of $\mathsf{Propose}(a)$.

▶ **Lemma 6.** *Let X be the set of even agents in G'. If $X \neq \emptyset$ then there exists $a \in X$ such that $\deg_{G'}(a) < k$.*

Proof. Suppose not, i.e., suppose every $a \in X$ satisfies $\deg_{G'}(a) = k$ (since $\deg_{G'}(a) \leq k$). Hence there are $k \cdot |X|$ edges incident to the set X in G'. Our invariant is that all neighbors in G' of a job b are tied in b's ranking. Since each tie has length at most k, it means that for any job b, $\deg_{G'}(b) \leq k$. Hence the set of neighbors of X in G' has size at least $(k \cdot |X|)/k$, i.e., $|\mathsf{Nbr}_{G'}(X)| \geq |X|$, where $\mathsf{Nbr}_{G'}(X)$ is the set of neighbors of X in G'.

However it follows from the definition of "even agents" that $|X| > |\mathsf{Nbr}_{G'}(X)|$. This is because every vertex in $\mathsf{Nbr}_{G'}(X)$ is odd (see Theorem 4) and so for every $v \in \mathsf{Nbr}_{G'}(X)$, there is a corresponding $M^*(v) \in X$, where M^* is any maximum matching in G'. There is also at least one vertex in X that is unmatched in M^*. Thus we get a contradiction that $|\mathsf{Nbr}_{G'}(X)| \geq |X| > |\mathsf{Nbr}_{G'}(X)|$. So there has to exist $a \in X$ with $\deg_{G'}(a) < k$. ◀

3.1 Correctness of our algorithm

Let M be the matching returned by Algorithm 1. Delete from M all edges incident to dummy jobs, so $M \subseteq E$. Let us first show that M is a stable matching in G, i.e., no edge *blocks* M.

▶ **Lemma 7.** *The matching M is stable in G.*

Proof. The termination condition of Algorithm 1 is that no agent is even in G'. Hence the original matching M (before deleting edges incident to dummy jobs) is A-perfect, i.e., it matches all agents. So there is no $(a,b) \in E$ such that b prefers a to its assignment in M and a is unmatched in M, however a never got to propose to b because $\deg_{G'}(a) = k$. Recall that agents add edges to G' in decreasing order of preference.

If (a,b) is in G' and a new edge (a,b') gets added to G' in the subroutine $\mathsf{Propose}(a)$, then a prefers b to b'. Since a is *even* prior to adding the edge (a,b'), it is easy to see that after adding the edge (a,b') to G', the agent a is either even or unreachable in G' (see footnote 3). Hence after the addition of (a,b'), the job b is odd/unreachable in G' (by Theorem 4).

Furthermore, the rank of b's neighborhood in G' never worsens as the algorithm progresses (this is justified below). Thus in M, which is a maximum matching in G', the job b is matched (by Theorem 4); moreover, it is matched to an agent at least as good as a.

We will next show that if an edge (a,b) gets deleted from G' then immediately after this deletion, b will be odd/unreachable in G' and b's neighbors in G' are ranked at least as good as a. There are three cases for the deletion of an edge (a,b) from G' in our algorithm.

[3] Prior to adding the new edge, a was even in G'. So all neighbors of a in the current G' will always be matched in any maximum matching, thus a has no *even* neighbor and hence a is not *odd*.

- *Case 1.* Suppose the edge (a, b) got deleted in line 1 in Algorithm 2. Then a, which is an even vertex in G', is going to make the next round of proposals. Just prior to the deletion of (a, b), the vertex b was *odd* in G' (since b has a neighbor a that is even). After the deletion of the edge (a, b), the vertex b either remains odd or it becomes unreachable. Also, by our invariant, all of b's neighbors in G' at this point are agents who are tied with a in b's preference list.
- *Case 2.* Suppose the edge (a, b) got deleted in line 6 in Algorithm 2. This is because b received a proposal from an agent a' that b prefers to a. In this case, a' will be b's current (and only) neighbor in G'. This makes b (and a') unreachable.
- *Case 3.* Suppose the edge (a, b) got deleted in line 12 in Algorithm 2. Just before the deletion of this edge, a was b's neighbor in G' and all of b's neighbors in G' were tied with a in b's preference list and have the same $\mathsf{rank}_{G'}$-value as $\mathsf{rank}_{G'}(a, b)$. As soon as b receives a proposal (say, from a') of the same rank as a and a better $\mathsf{rank}_{G'}$-value than $\mathsf{rank}_{G'}(a, b)$, b deletes edges to all its current neighbors and introduces the edge (a', b) in G'. So a' will be the only neighbor of b and this makes b (and a') unreachable in G'.

So for any $(a, b) \in E$, if a prefers b to its assignment in M then b likes its partner in M at least as much as a; hence (a, b) does not block M. Thus M is a stable matching in G. ◀

Bounding the unpopularity factor of M. For any pair of adjacent vertices v and w in G, let us define the function $\mathsf{vote}_v(w, M)$ as follows:

$$\mathsf{vote}_v(w, M) = \begin{cases} +1 & \text{if } v \text{ prefers } w \text{ to its assignment in } M; \\ -1 & \text{if } v \text{ prefers its assignment in } M \text{ to } w; \\ 0 & \text{otherwise.} \end{cases}$$

Thus $\mathsf{vote}_v(w, M)$ is v's vote for w versus its assignment in M and this value is 0 if v is indifferent between w and $M(v)$.

Edge labels. Let us label each edge (a, b) of $E \setminus M$ by $(\mathsf{vote}_a(b, M), \mathsf{vote}_b(a, M))$. Note that an edge labeled $(+1, +1)$ *blocks* M. We showed in Lemma 7 that no edge blocks M. So every edge in $E \setminus M$ gets a label in $\{(+1, -1), (-1, +1), (-1, -1), (\pm 1, 0)\}$. We would like to show that not too many edges are labeled $(+1, 0)$.

Let $\rho = \langle \ldots, e_1, f_1, e_2, f_2, \ldots, e_k, f_k, \ldots \rangle$ be any alternating path/cycle with respect to M, where each $e_i = (a_i, b_i)$ is in M. We will show an upper bound on the longest contiguous stretch of edges in $\rho \setminus M = \langle \ldots, f_1, f_2, \ldots, f_k, \ldots \rangle$ that can be labeled $(+1, 0)$.

▶ **Lemma 8.** *If all of $f_1, f_2, \ldots, f_{k-1}$ are labeled $(+1, 0)$ then f_k cannot be labeled $(+1, 0)$.*

Proof. Suppose all of $f_1, f_2, \ldots, f_{k-1}$ are labeled $(+1, 0)$. Since $f_1 = (a_1, b_2)$ is labeled $(+1, 0)$, b_2 is indifferent between a_1 and its partner a_2 in M while a_1 prefers b_2 to its partner b_1 in M (let $b_1 = d(a_1)$ if a_1 is unmatched in M).

If the edge (a_1, b_2) is present in G' then $\mathsf{rank}_{G'}(a_1, b_2) < \mathsf{rank}_{G'}(a_1, b_1)$ because a_1 prefers b_2 to b_1. Suppose the edge (a_1, b_2) is *not* present in G'. Let $\mathsf{rank}_{G'}(a_1, b_2)$ be the $\mathsf{rank}_{G'}$-value with which a_1 proposed to b_2 in the very last invocation of $\mathsf{Propose}(a_1)$. We claim $\mathsf{rank}_{G'}(a_2, b_2) < \mathsf{rank}_{G'}(a_1, b_2)$. This is because the edge (a_1, b_2) is missing in G' and the edge (a_2, b_2) is present in G' though b_2 is indifferent between a_1 and a_2. Thus in both cases, we can draw the conclusion that $\mathsf{rank}_{G'}(a_2, b_2) < \mathsf{rank}_{G'}(a_1, b_1)$:

- in the former case, it is $\mathsf{rank}_{G'}(a_2, b_2) = \mathsf{rank}_{G'}(a_1, b_2) < \mathsf{rank}_{G'}(a_1, b_1)$;
- in the latter case, it is $\mathsf{rank}_{G'}(a_2, b_2) < \mathsf{rank}_{G'}(a_1, b_2) \leq \mathsf{rank}_{G'}(a_1, b_1)$.

Consider any such 3-edge sequence $\langle e_i, f_i, e_{i+1}\rangle = b_i - a_i - b_{i+1} - a_{i+1}$ in ρ (see Fig. 1) where $1 \le i \le k-1$. Since $f_i = (a_i, b_{i+1})$ is labeled $(+1, 0)$, the same argument as the one above for $b_1 - a_1 - b_2 - a_2$ shows that $\mathsf{rank}_{G'}(a_{i+1}, b_{i+1}) < \mathsf{rank}_{G'}(a_i, b_i)$. Thus we have the following chain of strict inequalities: $\mathsf{rank}_{G'}(a_k, b_k) < \cdots < \mathsf{rank}_{G'}(a_1, b_1)$.

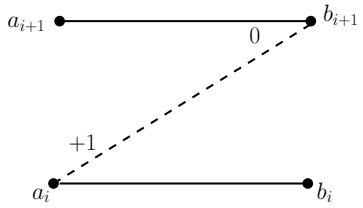

Figure 1 The dashed edge (a_i, b_{i+1}) is labeled $(+1, 0)$ and $\mathsf{rank}_{G'}(a_{i+1}, b_{i+1}) < \mathsf{rank}_{G'}(a_i, b_i)$.

We have $\mathsf{rank}_{G'}(a_1, b_1) \le k$ since $\mathsf{rank}_{G'}(a_1, b_1) \le \deg_{G'}(a_1) \le k$. So the chain of strict inequalities $\mathsf{rank}_{G'}(a_k, b_k) < \cdots < \mathsf{rank}_{G'}(a_1, b_1)$ implies that $\mathsf{rank}_{G'}(a_k, b_k) \le 1$. Hence $\mathsf{rank}_{G'}(a_k, b_k) = 1$, i.e., b_k is a_k's most favorite neighbor in the graph G'.

We need to show that the edge $f_k = (a_k, b_{k+1})$ is *not* labeled $(+1, 0)$. If a_k prefers b_k to b_{k+1} then f_k is labeled $(-1, *)$ and we are done. The non-trivial case is when a_k prefers b_{k+1} to b_k. The following claim is proved below.

▷ Claim 9. If a_k prefers b_{k+1} to b_k then b_{k+1} is matched in M to an agent better than a_k.

So if a_k prefers b_{k+1} to b_k then $\mathsf{vote}_{b_{k+1}}(a_k, M) = -1$ (by Claim 9). Hence $f_k = (a_k, b_{k+1})$ is labeled $(+1, -1)$. Thus the edge f_k is never labeled $(+1, 0)$. ◀

Proof. (of Claim 9) The agent a_k prefers b_{k+1} to b_k. Suppose the edge (a_k, b_{k+1}) is present in G_0. Then before proposing to b_k with $\mathsf{rank}_{G'}(a_k, b_k) = 1$, the agent a_k would have proposed to b_{k+1} with $\mathsf{rank}_{G'}(a_k, b_{k+1}) = 1$. Since (a_k, b_{k+1}) is present in G_0, note that a_k is as good as b_{k+1}'s neighbors in G' (wrt b_{k+1}'s ranking). Moreover, b_{k+1} would never have rejected a_k's proposal due to *poor* $\mathsf{rank}_{G'}$-value as the edge (a_k, b_{k+1}) has the best possible $\mathsf{rank}_{G'}$-value of 1. Thus if (a_k, b_{k+1}) is in G_0 then the edge (a_k, b_{k+1}) has to be in G' and so $\mathsf{rank}_{G'}(a_k, b_k)$ would be at least 2 since a_k prefers b_{k+1} to b_k. However $\mathsf{rank}_{G'}(a_k, b_k) = 1$.

So the edge (a_k, b_{k+1}) is *not* present in G_0. This means that b_{k+1} received one or more proposals from neighbors that it prefers to a_k. Hence b_{k+1} is matched in M to a neighbor that it prefers to a_k. ◁

▶ Remark 10. If ρ is an alternating cycle with respect to M then the proof of Lemma 8 shows that *all* the edges in $\rho \setminus M$ cannot be labeled $(+1, 0)$.

▶ **Lemma 11.** *The unpopularity factor of M is at most k.*

Proof. Let N be any matching in G. Consider $M \oplus N$ which is a set of alternating paths and alternating cycles. Let p be any alternating path in $M \oplus N$. Suppose p has even length, so $p = \langle e_1, f_1, \ldots, e_t, f_t\rangle$ where for each $i \in [t]$, the edge $e_i = (a_i, b_i) \in M$ and the edge $f_i = (a_i, b_{i+1}) \in N$. We have $p \setminus M = p \cap N = \langle f_1, \ldots, f_t\rangle$. Since b_{t+1}, i.e., the "job endpoint" of the last edge f_t in $p = \langle e_1, f_1, \ldots, e_t, f_t\rangle$, prefers to be matched rather than be unmatched (observe that M leaves b_{t+1} unmatched) and because no edge is labeled $(+1, +1)$, the edge f_t has to be labeled $(-1, +1)$.

Suppose p has odd length, so $p = \langle e_1, f_1, \ldots, e_t\rangle$. The final vertex of p (an endpoint of e_t) is an agent a_t matched in M but not in N, so a_t prefers M to N. The edge $f_{t-1} = (a_{t-1}, b_t)$ could be labeled $(+1, 0)$, however since a_t prefers M to N, the votes for N from the three

vertices a_{t-1}, b_t, a_t are $+1$, 0, and -1, respectively. Hence this case is equivalent to the case where p has even length, i.e., the final vertex of p is in B and the last edge is labeled $(-1, +1)$. Thus in the rest of the proof, we assume without loss of generality that p has even length.

Let us extract disjoint *maximal* contiguous segments $s_1, \ldots, s_r$ from $p \cap N = \langle f_1, \ldots, f_t \rangle$ such that for each i, every edge in $s_i = \langle f_{i_1}, f_{i_2} \ldots \rangle$ is labeled $(+1, 0)$ and every edge in $p \cap N$ that is labeled $(+1, 0)$ belongs to one of $s_1, \ldots, s_r$. We know from Lemma 8 that $|s_i| \leq k - 1$ for all $i \in [r]$. The maximality of each of the segments $s_1, \ldots, s_r$ along with our observation above that f_t is labeled $(-1, +1)$ implies the following:

- for each $i \in [r]$, the edge (call it $f_{i'}$) in $p \cap N = \langle f_1, \ldots, f_t \rangle$ that immediately follows the last edge in s_i has at least one "-1" in its label.

For each $i \in [r]$, let us map all the edges in s_i to this edge $f_{i'}$.

For every alternating path (similarly, alternating cycle) ρ in $M \oplus N$, as done above for the alternating path p, let us perform an analogous operation of extracting such maximal contiguous segments of $(+1, 0)$-labeled edges from $\rho \cap N$ and mapping all the edges in each such segment to the edge in $\rho \cap N$ that immediately follows this segment – this is an edge with a "-1" in its label. So every edge labeled $(+1, 0)$ in N gets mapped to an edge in N that has at least one -1 in its label. Moreover, at most $k - 1$ edges are mapped to any edge.

Our goal is to bound $\phi(N, M)$ = the number of vertices that prefer N to M. Every vertex that prefers N to M contributes $+1$ to the edge of N incident to it. We know from Lemma 7 that no edge is labeled $(+1, +1)$. Thus $\phi(N, M)$ = the number of edges in $N \setminus M$ labeled by one of $(+1, 0), (+1, -1), (-1, +1)$. We have shown that the number of edges labeled $(+1, 0)$ is at most $(k-1) \cdot$ (the number of edges labeled by one of $(+1, -1), (-1, +1), (-1, 0), (-1, -1)$). Moreover, the edge labels $(+1, -1)$ and $(-1, +1)$ have both "$+1$" and "-1" in them.

Hence we can conclude the following inequality on the labels of edges in $N \setminus M$: (the total number of $+1$'s in these labels) $\leq k \cdot$(the total number of -1's in these labels). Since $\phi(N, M)$ is the total number of $+1$'s in the labels of edges in $N \setminus M$ and $\phi(M, N)$ is at least the total number of -1's in these edge labels, we have $\phi(N, M) \leq k \cdot \phi(M, N)$. Since this holds for any matching N, we have $u(M) \leq k$. ◀

3.2 Running time of our algorithm

We now bound the total number of times the subroutine Propose gets called in Algorithm 1. Let $a \in A$. After any invocation of Propose(a), either $\deg_{G'}(a) = k$ or a is unreachable in G'. Let $\deg(a)$ be a's degree in G.

▷ Claim 12. The total number of invocations of the subroutine Propose(a) where $\deg_{G'}(a) = k$ at the end of this invocation is at most $k \cdot \deg(a)$.

Proof. Suppose $\deg_{G'}(a) = k$ at the end of an invocation of Propose(a). For Propose(a) to be called again, $\deg_{G'}(a)$ should be less than k. So between these two invocations of Propose(a), some edge (a, b) must have been deleted from G'. An edge $e = (a, b)$, once deleted from G' because b does not regard $\mathsf{rank}_{G'}(e)$ good enough, can get re-introduced in G' in line 13 or line 15 in a later invocation of Propose(a). However it has to be the case that the edge e now has a smaller value of $\mathsf{rank}_{G'}(e)$. Thus in the earlier invocation of Propose(a), the pair $(e, \mathsf{rank}_{G'}(e))$ with the older value of $\mathsf{rank}_{G'}(e)$ occurred in G' for the *last time*.

So let us charge the pair $(e, \mathsf{rank}_{G'}(e))$ for the invocation of Propose(a) where $(e, \mathsf{rank}_{G'}(e))$ occurs for the last time. Because a has at most k neighbors in G', we have $\mathsf{rank}_{G'}(e) \in [k]$ for any edge e in G'. Thus there are at most $k \cdot \deg(a)$ pairs $(e', \mathsf{rank}_{G'}(e'))$ where e' is any edge incident to a. Hence the total number of invocations of Propose(a) where $\deg_{G'}(a) = k$ at the end of this invocation is at most $k \cdot \deg(a)$. ◁

▷ Claim 13. The total number of invocations of the subroutine Propose(a) where a is unreachable in G' at the end of this invocation is at most $k \cdot \deg(a)$.

Proof. Suppose the vertex a is unreachable in G' at the end of an invocation of Propose(a). Let (a, b) be the last edge incident to a that got added to G' in this invocation of Propose(a). Observe that prior to adding the edge (a, b) to G', the vertex a is even in G'. So it is this edge (a, b) that made a unreachable in G'. We claim that the edge (a, b) is:

1. either a *new* edge that has been added to G' for the very first time[4]
2. or it is an *old* edge but with a smaller $\mathsf{rank}_{G'}$-value than the previous time it was added.

We argue that either (1) or (2) stated above must have occurred because none of the old edges incident to a along with their previous $\mathsf{rank}_{G'}$-values was good enough for a to be unreachable in G'. This is because for the subroutine Propose(a) to get invoked, the agent a has to be even in G'. Thus none of the old edges incident to a along with their previous $\mathsf{rank}_{G'}$-values was able to ensure a's *unreachability* in G'. So if at the end of this invocation of Propose(a), a is unreachable in G', then either an old edge with a new $\mathsf{rank}_{G'}$-value or a new edge must have been introduced in G' in this invocation of Propose(a).

So for every invocation of Propose(a) where a is unreachable in G' at the end of this invocation, there is a pair $(e, \mathsf{rank}_{G'}(e))$, where e is incident to a, that occurs in G' for the *first time*. We know that the number of pairs $(e, \mathsf{rank}_{G'}(e))$ is at most $k \cdot \deg(a)$. Thus the number of invocations of Propose(a) where a is unreachable in G' at the end of this invocation is at most $k \cdot \deg(a)$. Hence the claim follows. ◁

Since at the end of an invocation of Propose(a), either $\deg_{G'}(a) = k$ or a is unreachable in G', it follows that the total number of times Propose(a) gets called is $O(k \cdot \deg(a))$. Thus added up over all a in A, the total number of times the subroutine Propose gets called in Algorithm 1 is $O(\sum_{a \in A} k \cdot \deg(a)) = O(k \cdot m)$.

Running time of the subroutine Propose**.** As done in the implementation of the Gale-Shapley algorithm, the step where a job b deletes edges to worse-ranked neighbors in G_0 is implemented in a delayed manner. That is, before an agent a proposes to any neighbor b, it first checks if the edge (a, b) is actually present in G_0. This can be easily implemented such that the total time taken in Algorithm 1 by all the steps to delete edges from G_0 is $O(m)$.

Regarding the other steps in the subroutine Propose, we will always maintain a maximum matching M in the current G'. We will mark vertices as even/odd/unreachable – given a maximum matching in G', this takes time linear in the size of G'. The number of edges in G' is at most $k \cdot |A| = k \cdot n$ since every $a \in A$ has degree at most k in G'.

For the subroutine Propose(a) to be invoked, the agent a has to be even in G'. Hence there is an alternating path ρ between a and some agent left unmatched in M. We will update M to $M \oplus \rho$ so that a is unmatched in the resulting M. Recall that the first step of the subroutine Propose(a) deletes all edges incident to a in G'; subsequently, a proposes to its neighbors in G_0 one-by-one and adds some edges to G'.

Adding an edge between a and a neighbor that is odd/unreachable in G' maintains the property that M is a maximum matching in G'. In order to get a larger matching in G', we need to add an edge between a and a neighbor b that is *even* in G'. The only way b can be

[4] Note that once the edge $(a, d(a))$ gets added to G', a remains unreachable in G' henceforth – so if a is even in G' then there is always at least one edge incident to a in G_0 that is not in G'.

even in G' is for b to be *isolated* just before the addition of the edge (a, b) to G'.[5] So either b never received any proposal till now in Algorithm 1 or b may have had neighbors in G', however upon receiving a's proposal in Propose(a), the vertex b had to delete all its incident edges in G' and then the edge (a, b) got added to G'.

Since b was isolated just before adding (a, b), adding this edge makes a *unreachable* in G' (see footnote 3) and the subroutine Propose(a) ends. We will update M: the edge (a, b) is added to M and if b was matched earlier, then this old edge (which is no longer in G') is deleted from M. Thus we maintain a maximum matching M in G'.

At the end of the subroutine Propose, we will search for an agent that is even in G' and with degree less than k. If such an agent is found, then the while-loop of Algorithm 1 continues; else Algorithm 1 terminates. Searching for such an agent takes time linear in the size of G'. Thus the running time of any invocation of Propose is $O(k \cdot n)$. Since there are $O(k \cdot m)$ invocations of Propose, Lemma 14 follows.

▶ **Lemma 14.** *The running time of Algorithm 1 is $O(k^2 \cdot mn)$.*

Theorem 2 stated in Section 1 follows from Lemma 11 and Lemma 14.

4 Maximum matchings and approximate popularity

Our goal in this section is to find a maximum matching M in $G = (A \cup B, E)$ such that $\phi(N, M) \leq 2k \cdot \phi(M, N)$ for any maximum matching N in G. Recall that G is a bipartite graph with one-sided ties of length at most k.

We will construct a new graph $H = (A \cup B, E^*)$ where every edge in G is replicated n times, recall $|A| = n$. Thus $E^* = \cup_{e \in E}\{e_1, \ldots, e_n\}$, i.e., for every $e \in E$, there are n copies $e_1, \ldots, e_n$ in E^*. So H is a multigraph. Due to the presence of parallel edges in H, every vertex v in H has preferences over its incident edges (rather than its neighbors).

Let the preference order of a vertex v over its incident edges in G be $e \succeq e' \succeq e'' \succeq \cdots$. In the graph H, the edges $e_1, \ldots, e_n$ (i.e., n copies of e) along with $e'_1, \ldots, e'_n$ (i.e., n copies of e') and $e''_1, \ldots, e''_n$ (i.e., n copies of e'') and so on are incident to v.

- If $v \in A$ then the preference order of v over its incident edges in H is as given below (recall that vertices in A have strict rankings in G):

$$\underbrace{e_1 \succ e'_1 \succ e''_1 \succ \cdots}_{\text{subscript 1 edges}} \succ \underbrace{e_2 \succ e'_2 \succ e''_2 \succ \cdots}_{\text{subscript 2 edges}} \succ \cdots \succ \underbrace{e_n \succ e'_n \succ e''_n \succ \cdots}_{\text{subscript } n \text{ edges}}.$$

  Vertices of A have strict rankings in H as well. For $i < j$, where $i, j \in [n]$, $v \in A$ prefers any subscript i edge incident to it to any subscript j edge incident to it. For any i, within the set of subscript i edges, v's preference order among $e_i, e'_i, e''_i, \ldots$ is as per v's preference order over its incident edges $e, e', e'', \ldots$ in G.

- If $v \in B$ then the preference order of v over its incident edges in H is as given below:

$$\underbrace{e_n \succeq e'_n \succeq e''_n \succeq \cdots}_{\text{subscript } n \text{ edges}} \succ \underbrace{e_{n-1} \succeq e'_{n-1} \succeq e''_{n-1} \succeq \cdots}_{\text{subscript } n-1 \text{ edges}} \succ \cdots \succ \underbrace{e_1 \succeq e'_1 \succeq e''_1 \succeq \cdots}_{\text{subscript 1 edges}}.$$

  Thus for $i < j$, where $i, j \in [n]$, $v \in B$ prefers any subscript j edge incident to it to any subscript i edge incident to it. For any i, within the set of subscript i edges, v's preference order over $e_i, e'_i, e''_i, \ldots$ is as per v's original preference order. So if v is indifferent between e and e' in G then v is indifferent between e_i and e'_i for every $i \in [n]$.

[5] For an *unisolated* b to be even in G', its neighbors have to be *odd* and no agent is ever odd in G'. Recall that as soon as the agent a becomes unreachable in Propose(a), the subroutine Propose(a) ends.

Thus in the graph H, while vertices in B prefer higher subscript edges, vertices in A prefer lower subscript edges. Analogous to Section 3, we have the graph H_0 where $H_0 = (A \cup B_0, E_0^*)$, $B_0 = B \cup \{d(a) : a \in A\}$, and $E_0^* = E^* \cup \{(a, d(a)) : a \in A\}$. For any $a \in A$, the edge $(a, d(a))$ is the worst ranked edge incident to a in H_0.

The algorithm. Let us call Algorithm 1 in the graph H to construct the subgraph H' of H_0. As before, in the subroutine Propose, any job b, upon receiving a proposal along an edge (say, e_i) deletes from H_0 all the edges that b ranks worse than e_i. In particular, all edges e'_j incident to b where $j < i$ get deleted from H_0.

Algorithm 1 returns a maximum matching M^* in H'. Let M be the corresponding matching in G, i.e., M is essentially the same as M^*, however edges incident to dummy jobs are pruned from M^* and the subscripts of edges in M^* are ignored in M.

The number of edges in H is $m \cdot n$ and every tie in H is also a tie in G, so it has length at most k. Hence the running time of Algorithm 1 in H is $O(k^2 \cdot (mn)n) = O(k^2 \cdot mn^2)$. Lemma 15 and Lemma 16 prove the correctness of our algorithm. Thus Theorem 3 stated in Section 1 follows.

▶ **Lemma 15.** *M is a maximum matching in G.*

Proof. We will show there is no augmenting path with respect to M in G. Let $a_0 \in A$ be a vertex left unmatched in M. Then in the graph H', the edge $(a_0, d(a_0))$ is in M^*. Since $(a_0, d(a_0))$ is the worst ranked edge incident to a_0, this means that for every edge e_i, in particular, for every subscript n edge e_n incident to a_0, either a_0 proposed along this edge e_n to the other endpoint (call it b_1) or b_1 had already deleted e_n from H_0 since it received a proposal along an edge better than e_n. Note that any edge better than e_n has to be a subscript n edge. The fact that a_0 added the edge $(a_0, d(a_0))$ to the subgraph H' implies that a_0 was *even* in H' before adding $(a_0, d(a_0))$. So at the end, every neighbor of a_0 is odd/unreachable in H'. Hence any neighbor b_1 of a_0 has to be matched along a subscript n edge in M^*.

Suppose $(a_1, b_1) \in M$. By the argument in the above paragraph, we know that it is the subscript n copy of this edge (a_1, b_1) that is present in M^*. Recall that any agent proposes or adds edges to H' in decreasing order of preference and every agent prefers lower subscript edges to higher subscript edges. So a_1 must have proposed along every subscript $(n-1)$ edge incident to it that was not yet deleted from H_0. Furthermore, any neighbor b_2 of a_1 that deleted the subscript $(n-1)$ copy of the edge (a_1, b_2) from H_0 has to be matched along a subscript $\geq (n-1)$ edge in M^*. Thus any neighbor of a_1 has to be matched along a subscript $\geq (n-1)$ edge in M^*.

Continuing this argument, any augmenting path ρ wrt M that starts with a_0 looks as follows (the minimum possible subscripts of these edges in M^* are written below them):

$$a_0 - \underbrace{b_1 - a_1}_{\text{subscript } n} - \underbrace{b_2 - a_2}_{\text{sub. } n-1} - \underbrace{b_3 - a_3}_{\text{sub. } n-2} - \cdots \underbrace{b_i - a_i}_{\text{sub. } n-i+1} - \cdots - \underbrace{b_n - a_n}_{\text{subscript } 1} - b.$$

Finally, the path ρ has to reach a job b that never received a proposal in H', in other words, b is isolated in H'. So every neighbor (say, a') of b has to be matched along a subscript 1 edge since a' never proposed along the subscript 1 edge e'_1 where $e' = (a', b)$. In our augmenting path ρ, any agent matched along a subscript 1 edge has to be labeled a_t where $t \geq n$. This means ρ includes at least $n+1$ agents $a_0, a_1, \ldots, a_n$. However this is impossible as $|A| = n$. Thus there is no augmenting path with respect to M. Equivalently, M is a maximum matching in G. ◀

▶ **Lemma 16.** *For any maximum matching N in G, $\phi(N, M) \leq 2k \cdot \phi(M, N)$.*

Before we prove Lemma 16, let us recall the following preliminaries from [17]. Let us partition $A = A_1 \cup \cdots \cup A_n$ and $B = B_1 \cup \cdots \cup B_n$ as follows:

- For every subscript i edge $e_i = (a, b)$ in M^*, add a to A_i and b to B_i.
- Add the vertices in A that are unmatched in M to A_n.
- Add the vertices in B that are unmatched in M to B_1.

It follows from the definition of the sets A_i, B_i that $M \subseteq \cup_{i=1}^{n}(A_i \times B_i)$. It will be useful to visualize these subscripts as *levels*. So $A = A_1 \cup \cdots \cup A_n$ is a partition of A into n levels where vertices in A_i are at level i and similarly, $B = B_1 \cup \cdots \cup B_n$ is a partition of B into n levels where vertices in B_i are at level i. Edges in $A_i \times B_j$ where $i < j$ are said to move *upwards* and edges in $A_i \times B_j$ where $i > j$ are said to move *downwards*. Claim 17 (from [17]) shows there is no *steep* downwards edge.

▷ Claim 17. There is no edge between A_i and B_j where $i \geq j + 2$.

Proof. Let $a \in A_i$. We need to show a has no neighbor in B_j where $j \leq i - 2$. This argument was seen in the proof of Lemma 15. Let us consider two cases.

1. a is matched in M: Since $a \in A_i$, a proposed along all the subscript $i - 1$ edges incident to it in H_0 (that were not yet deleted) but these proposals were rejected by its neighbors. Since vertices in B prefer higher subscript edges to lower subscript edges, this means that all neighbors of a are matched along $\geq i - 1$ subscript edges. Thus a has no neighbor in B_j where $j \leq i - 2$.
2. a is not matched in M: Here $a \in A_n$ and as argued in the proof of Lemma 15, $(a, d(a))$ is in M^* and all neighbors of a are in B_n. So a has no neighbor in B_j where $j \leq n - 1$. ◁

As done in Section 3, let us label edges in $E \setminus M$ by $(\mathsf{vote}_a(b, M), \mathsf{vote}_b(a, M))$. Claim 18 stated below tells us that no downwards edge can have "+1" in its label.

▷ Claim 18. For $2 \leq j \leq n$, any edge in $A_j \times B_{j-1}$ is labeled either $(-1, -1)$ or $(-1, 0)$.

Proof. Let $e = (a, b) \in A_j \times B_{j-1}$. We need to show that e is labeled either $(-1, -1)$ or $(-1, 0)$. The job b is matched in M^* along a subscript $(j - 1)$ edge e'_{j-1}. Recall that any job prefers higher subscript edges to lower subscript edges, so if a had proposed to b along e_j then b would have accepted the proposal. Thus we can conclude that a is matched along an edge e''_j that it prefers to e_j, i.e., a prefers its partner in M to b. Thus $\mathsf{vote}_a(b, M) = -1$.

Since agents prefer lower subscript edges to higher subscript edges, a proposed along e_{j-1} before proposing along the edge e''_j. Since b did not accept the proposal along e_{j-1}, we can conclude that b does not prefer e_{j-1} to the edge e'_{j-1} along which it is matched in M^*. Thus $\mathsf{vote}_b(a, M) \in \{0, -1\}$. ◁

▷ Claim 19. Let $e \in (A_j \times B_j) \setminus M$ where $j \in [n]$. Then e cannot be labeled $(+1, +1)$.

The proof of Claim 19 is the same as the proof of Lemma 7. Note that Claims 17-19 imply that any edge labeled $(+1, +1)$ has to be an *upwards* edge.

Proof. (of Lemma 16) Let $\rho = \langle \ldots, e_1, f_1, e_2, f_2, \ldots, \rangle$ be any alternating path/cycle with respect to M where the e-edges are in M. We will not be able to claim as done in Lemma 8 that the longest contiguous stretch of edges $\langle f_1, f_2, \ldots \rangle$ labeled $(+1, 0)$ in $\rho \setminus M$ is at most $k - 1$. Now $\rho \setminus M$ can admit longer contiguous stretches of *positively labeled* edges, i.e., those labeled $(+1, 0)$ or $(+1, +1)$, by using upwards edges, i.e., edges in $\cup_{i<j}(A_i \times B_j)$, interspersed with edges in $\cup_i(A_i \times B_i)$.

Let $\rho \setminus \{\text{upwards/downwards edges}\} = \rho_1 \cup \cdots \cup \rho_t$. For any $i \in [t]$, ρ_i consists of edges in $A_j \times B_j$ for some $j \in [n]$. Let $\rho_i \setminus M = \langle \ldots, f_{i_1}, f_{i_2}, \ldots, f_{i_{k-1}}, f_{i_k}, \ldots \rangle$. We have the following claim whose proof is the same as the proof of Lemma 8.

▷ Claim 20. If all of $f_{i_1}, f_{i_2}, \ldots, f_{i_{k-1}}$ are labeled $(+1, 0)$ then f_{i_k} cannot be labeled $(+1, 0)$.

Recall that we need to compare M only against maximum matchings. Let ρ be any *relevant* alternating path/cycle ρ wrt M, i.e., ρ occurs in $M \oplus N$ where N is a maximum matching. There are two cases here.

(1) ρ is an alternating cycle. The crucial observation is that the number of downwards edges in ρ is at least the number of upwards edges in ρ. This is because ρ is a cycle and there are no *steep* downwards edges (by Claim 17).

Recall that $\rho \setminus \{\text{upwards/downwards edges}\} = \rho_1 \cup \cdots \cup \rho_t$. Each of the edges in $\rho_i \setminus M$ is in $A_j \times B_j$ for some $j \in [n]$. We know from Claim 19 that no edge in $\rho_i \setminus M$ is labeled $(+1, +1)$. Furthermore, there can be at most $k-1$ contiguous stretch of edges labeled $(+1, 0)$ in $\rho_i \setminus M$ (by Claim 20).

For any maximal contiguous segment of edges labeled $(+1, 0)$ in $\rho_i \setminus M = \langle \ldots, f_{i_1}, f_{i_2}, \ldots \rangle$, except possibly for the last such segment (call it ℓ_i), the maximality of each segment implies that the edge that immediately follows this segment in $\rho_i \setminus M$ has at least one "-1" in its label. Analogous to Lemma 11, this "-1" will *pay* for the $+1$'s in this segment.

For the last segment ℓ_i in $\rho_i \setminus M$, we have to find a "-1" to pay for the $+1$'s in this segment. Note that there might be no edge in $\rho_i \setminus M$ that follows the last segment ℓ_i. This is because the edge in $\rho \setminus M$ that immediately follows ℓ_i moves to another level – so it is either a *downwards* edge or an *upwards* edge. Any downwards edge is labeled $(-1, 0)$ or $(-1, -1)$ (by Claim 18) while an upwards edge might be labeled $(+1, +1)$. Thus a downwards edge is a good edge with at least one "-1" in its label while an upwards edge is possibly a bad edge with up to two $+1$'s in its label. Summarizing,

- we have to pay for the $+1$'s in the labels of upwards edges and
- we have to pay for the *last-segments* $\ell_1, \ldots, \ell_t$ in $\rho_1, \ldots, \rho_t$, respectively.

We have t last-segments and some $s \leq t$ upwards edges. Our goal is to show that there are enough downwards edges in ρ so that $2k$ times their number is an upper bound on the total number of $+1$'s in the upwards edges and in the last-segments.

Consider the following list: $\ell_1, f'_1, \ell_2, f'_2, \ldots, \ell_t, f'_t$, where $\ell_1, \ldots, \ell_t$ are all the last-segments and for $i \in [t]$, f'_i is the upwards/downwards edge in $\rho \setminus M$ that immediately follows $\rho_i \setminus M$. Each edge in ℓ_i is labeled $(+1, 0)$ while the edge f'_i (if it is an upwards edge) could be labeled $(+1, +1)$.

Let $F = \{f'_1, \ldots, f'_t\}$ and let $X \subseteq F$ be the set of downwards edges and let $Y \subseteq F$ be the set of upwards edges. We know that the number of downwards edges is at least the number of upwards edges, i.e., $|X| \geq |Y| = s$ and $|X| + |Y| = t$. So $t - s \geq s$, i.e., $s \leq t/2$. For convenience, let us assume that $X = \{f'_1, \ldots, f'_{t-s}\}$ and $Y = \{f'_{t-s+1}, \ldots, f'_t\}$.

Our charging mechanism is as follows:

- For each $1 \leq i \leq t - s$, the "-1" in edge f'_i will pay for all the $+1$'s in the segment ℓ_i.
- For each $1 \leq i \leq s$, the "-1" in f'_i will also pay for all the $+1$'s in the segment ℓ_{t-s+i} and moreover, for the $+1$'s in the edge f'_{t-s+i}.

Since there are at most $(k-1)$ $+1$'s in any ℓ_j and at most two $+1$'s in any upwards edge, it follows that the total number of $+1$'s that any "-1" pays for is at most $2(k-1) + 2 = 2k$.

(2) ρ is an even length alternating path. So ρ has one endpoint that is unmatched in M. There are two subcases here based on whether the unmatched endpoint is in A or in B.

- Suppose the unmatched endpoint of ρ is in A. Recall that any vertex $a \in A$ that is unmatched in M is in the set A_n. Since ρ has even length, the final vertex in ρ is a matched agent $a' \in A_i$ for some $i \leq n$. Because there are no steep downwards edges, the number of downwards edges in ρ is at least the number of upwards edges.
  Thus the following crucial property – the number of downwards edges in ρ is at least the number of upwards edges – used in case (1) holds in this subcase as well. Now the rest of the argument that the total number of $+1$'s in ρ is at most $2k \cdot$ (the number of -1's in ρ) is the same as given in case (1).
- Suppose the unmatched endpoint of ρ is in B. Recall that any vertex $b \in B$ that is unmatched in M is in the set B_1. Observe that any neighbor a of an unmatched $b \in B$ is in A_1. This is because a never proposed along the subscript 1 edge corresponding to (a, b) and this means that a is matched along a subscript 1 edge.
  The other endpoint of ρ is a vertex $b' \in B$ that is matched in M. Thus the final vertex b' is in B_i for some $i \geq 1$ and its partner in M (call it a') is in A_i. Hence the alternating path ρ traverses from $a' \in A_i$ where $i \geq 1$ to $a \in A_1$. Since there are no steep downwards edges, the crucial property that the number of downwards edges in ρ is at least the number of upwards edges holds here as well. The rest of the argument that the total number of $+1$'s in ρ is at most $2k \cdot$ (the number of -1's in ρ) is the same as in case (1).

For any maximum matching N, the symmetric difference $M \oplus N$ is a collection of alternating cycles and alternating paths of even length. For any such alternating path or alternating cycle ρ, we know that (the number of $+1$'s in ρ) $\leq 2k \cdot$ (the number of -1's in ρ). Equivalently, $\phi(M \oplus \rho, M) \leq 2k \cdot \phi(M, M \oplus \rho)$. Hence $\phi(N, M) \leq 2k \cdot \phi(M, N)$. ◀

5 Conclusions and open problems

We showed that any bipartite graph $G = (A \cup B, E)$ where vertices in A have strict rankings over their neighbors and vertices in B have weak rankings with ties of length at most k admits a stable matching M with unpopularity factor at most k. So no matching N can win more than $k/(k+1)$-fraction of votes in a head-to-head election against M, where vertices are voters. We showed a polynomial time algorithm to find such a matching M. It is easy to show instances with one-sided ties of length at most k where every matching has unpopularity factor at least $k-1$. Thus our bound on unpopularity factor is almost tight.

An open problem is to extend the results shown here to *two-sided* ties of bounded length, i.e., given an instance $G = (A \cup B, E)$ where all vertices are allowed to have weak rankings with ties of length at most k, is there always a matching in G with unpopularity factor at most k? Furthermore, is there always such a stable matching? Is it easy to find one?

References

1 F. Bauckholt, K. Pashkovich, and L. Sanitá. On the approximability of the stable marriage problem with one-sided ties. `arXiv:1805.05391`.

2 S. Bhattacharya, M. Hoefer, C.-C. Huang, T. Kavitha, and L. Wagner. Maintaining near-popular matchings. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)(II)*, pages 504–515, 2015.

3 P. Biro, R. W. Irving, and D. F. Manlove. Popular matchings in the marriage and roommates problems. In *Proceedings of the 7th International Conference on Algorithms and Complexity (CIAC)*, pages 97–108, 2010.

4 P. Biro, D. F. Manlove, and S. Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411:1828–1841, 2010.

5 Á. Cseh. Popular matchings. Trends in Computational Social Choice, Ulle Endriss (ed.), 2017.

6 Á. Cseh, C.-C. Huang, and T. Kavitha. Popular matchings with two-sided preferences and one-sided ties. *SIAM Journal on Discrete Mathematics*, 31(4):2348–2377, 2017.

7 A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs. *Canadian Journal of Mathematics*, 10:517–534, 1958.

8 Y. Faenza and T. Kavitha. Quasi-popular matchings, optimality, and extended formulations. *Mathematics of Operations Research*, 47(1):427–457, 2022.

9 Y. Faenza, T. Kavitha, V. Powers, and X. Zhang. Popular matchings and limits to tractability. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2790–2809, 2019.

10 D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962.

11 P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Science*, 20:166–173, 1975.

12 S. Gupta, P. Misra, S. Saurabh, and M. Zehavi. Popular matching in roommates setting is NP-hard. *ACM Transactions on Computation Theory*, 13(2):1–20, 2021.

13 C.-C. Huang and T. Kavitha. Near-popular matchings in the roommates problem. *SIAM Journal on Discrete Mathematics*, 27(1):43–62, 2013.

14 C.-C. Huang and T. Kavitha. Improved approximation algorithms for two variants of the stable marriage problem with ties. *Mathematical Programming*, 154(1):353–380, 2015.

15 K. Iwama, S. Miyazaki, and N. Yamauchi. A 1.875-approximation algorithm for the stable marriage problem. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 288–297, 2007.

16 K. Iwama, S. Miyazaki, and N. Yamauchi. A 25/17-approximation algorithm for the stable marriage problem with one-sided ties. *Algorithmica*, 68(3):758–775, 2014.

17 T. Kavitha. A size-popularity tradeoff in the stable marriage problem. *SIAM Journal on Computing*, 43(1):52–71, 2014.

18 Z. Király. Better and simpler approximation algorithms for the stable marriage problem. *Algorithmica*, 60(1):3–20, 2011.

19 C.-K. Lam and C. G. Plaxton. A $(1 + 1/e)$-approximation algorithm for maximum stable matching with one-sided ties and incomplete lists. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2823–2840, 2019.

20 L. Losász and M. D. Plummer. *Matching theory*. North-Holland, Mathematics Studies 121, 1986.

21 D. F. Manlove and R. W. Irving. Approximation algorithms for hard variants of the stable marriage and hospitals/residents problems. *Journal of Combinatorial Optimization*, 16:279–292, 2008.

22 M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of the 8th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 593–604, 2008.

23 W. R. Pulleyblank. Chapter 3, matchings and extensions. The Handbook of Combinatorics, R.L. Graham, M. Grötschel, and L. Lovasz (ed.), 1995.

24 S. Ruangwises and T. Itoh. Unpopularity factor in the marriage and roommates problems. *Theory of Computing Systems*, 65(3):579–592, 2021.

25 M. Soldner. Optimization and measurement in humanitarian operations: Addressing practical needs. PhD thesis, Georgia Institute of Technology, 2014.

26 A.C. Trapp, A. Teytelboym, A. Martinello, T. Andersson, and N. Ahani. Placement optimization in refugee resettlement. Working paper, 2018.