Popularity at Minimum Cost

Telikepalli Kavitha[1], Meghana Nasre[2], Prajakta Nimbhorkar[3]

[1] Tata Institute of Fundamental Research, India. kavitha@tcs.tifr.res.in
[2] Indian Institute of Science, India. meghana@csa.iisc.ernet.in
[3] The Institute of Mathematical Sciences, India. prajakta@imsc.res.in

Abstract. We consider an extension of the *popular matching* problem in this paper. The input to the popular matching problem is a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$, where $\mathcal{A}$ is a set of people, $\mathcal{B}$ is a set of items, and each person $a \in \mathcal{A}$ ranks a subset of items in an order of preference, with ties allowed. The popular matching problem seeks to compute a matching M^* between people and items such that there is no matching M where more people are happier with M than with M^*. Such a matching M^* is called a popular matching. However there are simple instances where no popular matching exists.

Here we consider the following natural extension to the above problem: associated with each item $b \in \mathcal{B}$ is a non-negative price $\mathsf{cost}(b)$, that is, for any item b, new copies of b can be added to the input graph by paying an amount of $\mathsf{cost}(b)$ per copy. When G does not admit a popular matching, the problem is to "augment" G at minimum cost such that the new graph admits a popular matching. We show that this problem is NP-hard; in fact, it is NP-hard to approximate it within a factor of $\sqrt{n_1}/2$, where n_1 is the number of people. This problem has a simple polynomial time algorithm when each person has a preference list of length at most 2. However if we consider the problem of *constructing* a graph at minimum cost that admits a popular matching that matches all people, then even with preference lists of length 2, the problem becomes NP-hard. On the other hand, when the number of copies of each item (i.e., its capacity) is *fixed*, we show that the problem of computing a minimum cost popular matching or deciding that no popular matching exists can be solved in $O(mn_1)$ time.

1 Introduction

The *popular matching* problem deals with matching people to items, where each person ranks a subset of items in an order of preference, with ties allowed. The input is a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ where $\mathcal{A}$ is the set of people, $\mathcal{B}$ is the set of items and the edge set $E = E_1 \cup \cdots \cup E_r$ (E_i is the set of edges of rank i). For any $a \in \mathcal{A}$, we say a prefers item b to item b' if the rank of edge (a, b) is smaller than the rank of edge (a, b'). If the ranks of (a, b) and (a, b') are the same, then a is indifferent between b and b'. The goal is to match people with items in an *optimal* manner, where the definition of optimality will be a function of the preferences expressed by the elements of $\mathcal{A}$. The problem of computing such an optimal matching is a well studied problem and several notions of optimality have been considered so far; for instance, pareto-optimality [1], rank-maximality [4], and fairness.

One criterion that does not use the absolute values of the ranks is the notion of *popularity*. Let $M(a)$ denote the item to which an applicant a is matched in a matching M. We say that a person 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 $M'(a)$.

Definition 1. *M is* more popular than *M', denoted by $M \succ M'$, if the number of people who prefer M to M' is higher than those that prefer M' to M. A matching M^* is* popular *if there is no matching that is more popular than M^*.*

Popular matchings were first introduced by Gardenfors [3] in the context of stable matchings. Popular matchings can be considered to be stable as no majority vote of people can force a migration to another matching and also, popularity is based on *relative* ranking rather than the actual ranks used by people. Hence popularity is a desirable notion of optimality in the domain of one-sided preferences.

On the flip side, popularity does not provide a complete answer since there exist simple instances that do not admit any popular matching. An example is the following: let $\mathcal{A} = \{a_1, a_2, a_3\}$, $\mathcal{B} = \{b_1, b_2, b_3\}$,

and each person a_i, for $i = 1, 2, 3$, prefers b_1 to b_2, and b_2 to b_3. Consider the three symmetrical matchings $M_1 = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$, $M_2 = \{(a_1, b_3), (a_2, b_1), (a_3, b_2)\}$ and $M_3 = \{(a_1, b_2), (a_2, b_3), (a_3, b_1)\}$. None of these matchings is popular, since $M_1 \prec M_2$, $M_2 \prec M_3$, and $M_3 \prec M_1$. Abraham et al. [2] designed efficient algorithms for determining if a given instance admits a popular matching and computing one, if it exists.

The fact that popular matchings do not always exist has motivated several extensions to the popular matching problem. McCutchen [10] considered the problem of computing a *least unpopular* matching; he considered two measures of unpopularity and showed that computing a matching that minimized either of these measures is NP-hard. Kavitha et al. [5] generalized the notion of popularity to *mixed matchings* or probability distributions over matchings and showed that a popular mixed matching always exists.

Our problem. Here we consider another natural generalization to the popular matching problem: *augment* the input graph G such that the new graph admits a popular matching. Our input consists of $G = (\mathcal{A} \cup \mathcal{B}, E)$ and a function $\mathsf{cost} : \mathcal{B} \to \mathbb{R}^+$, where $\mathsf{cost}(b)$ for any $b \in \mathcal{B}$ is the cost of making a new copy of item b. The set $\mathcal{B}$ is a set of items, say books or DVDs, and new copies of any $b \in \mathcal{B}$ can be obtained by paying $\mathsf{cost}(b)$ for each new copy of b. There is no restriction on the number of copies of any item that can be made. The only criterion that we seek to optimize is the total cost of augmenting G.

Going to back to the earlier example on 3 people and 3 items that did not admit a popular matching, it is easy to show that by making a new copy of either b_1 or b_2, the resulting graph admits a popular matching.[4] Our starting graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ comes for free, every *addition* that we make to G comes at a price and our goal is to make these additions such that the new graph admits a popular matching and the total cost of additions is minimized. We call this the *min-cost augmentation* problem.

A related problem. A related problem is the following: we do not have a starting graph G. We are given a set $\mathcal{A}$ of people and their preference lists over a universe U of items where each item $b \in U$ has a price $\mathsf{cost}(b) \geq 0$ associated with it. The problem is to "construct" an input graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ where $\mathcal{B}$ is a multiset of some elements in U such that G admits a popular matching and the cost of constructing G, that is, $\sum_{b \in \mathcal{B}} \mathsf{cost}(b)$, is as small as possible. Here we also have an extra condition that the popular matching should leave no person unmatched, otherwise we have a trivial solution of $\mathcal{B} = \emptyset$. We call this problem the *min-cost popular instance* problem.

The above problem can also be regarded as a "gift buying" problem. Each person in $\mathcal{A}$ has a preference list over gifts that she would like to receive. The problem is to buy a gift for each person in $\mathcal{A}$ with the total cost as small as possible and assign each person a gift such that this assignment is popular. That is, there is no reassignment of gifts that makes more people happier. This is the min-cost popular instance problem.

1.1 Our Results

We show the following results in this paper:

1. The min-cost popular instance problem is NP-hard, even when each preference list has length at most 2 (i.e., every applicant has a top choice item and possibly, a second choice item).
2. The min-cost augmentation problem has a polynomial time algorithm when each preference list has length at most 2 .
3. The min-cost augmentation problem is NP-hard for general lists. In fact, it is NP-hard to approximate to within a factor of $\sqrt{n_1}/2$, where n_1 is the number of people.

All our NP-hardness results hold even when preference lists are derived from a *master list*. A master list is a total ordering of the items according to some global objective criterion. Thus if b_1 precedes b_2 in the master list and if a person a has both b_1 and b_2 in her list, then it has to be the case that b_1 precedes b_2 in a's list.

We would like to contrast the NP-hardness of the min-cost augmentation problem with the problem of determining a popular matching with variable job capacities [7]. In this problem the input is a

[4] In order to minimize the cost, we will make a new copy of that item in $\{b_1, b_2\}$ which has lower cost.

graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ where $\mathcal{A}$ is a set of people and $\mathcal{B}$ is a set of jobs, we are also given a list $\langle c_1, \ldots, c_{|\mathcal{B}|} \rangle$ denoting upper bounds on the capacities of each job. The problem is to determine if there exists $(x_1, \ldots, x_{|\mathcal{B}|})$ such that for each i, setting the capacity of i-th job to x_i, where $1 \leq x_i \leq c_i$, enables the resulting graph to admit a popular matching. This problem was shown to be NP-hard in [7]; however note that here the capacity of *each* job has an upper bound. Instead, if we only had to maintain an overall upper bound on the total increase of capacities rather than individual upper bounds, a simple polynomial time algorithm solves this problem [7].

In the min-cost augmentation problem recall that there is no upper bound on the amount that we can spend on a particular item. What we seek to optimize is the overall cost and this problem is NP-hard. Note that when each item has the same cost, then this problem can be solved in polynomial time (using the above algorithm from [7]). However when the costs come from $\{1, 2\}$ the problem becomes NP-hard.

The NP-hardness results for the min-cost augmentation/min-cost popular instance problems are because the number of copies of each of the items need to be determined so as to ensure the existence of a popular matching at minimum cost. Let $\mathsf{cap}(b)$ for any item $b \in \mathcal{B}$ denote the number of copies of item b in our graph G. We now consider the following problem: each $b \in \mathcal{B}$ has a *fixed* capacity $\mathsf{cap}(b)$ and let the cost of a matching M be the sum of costs of items that are matched in M (we have to pay a cost of $k.\mathsf{cost}(b)$ if a capacity of k, where $k \leq \mathsf{cap}(b)$, of item b is used in M). Our final result is a polynomial time algorithm for the *min-cost popular matching* problem which we define below.

- The *min-cost popular matching* problem is to determine if G admits a popular matching or not and if so, to compute the one with minimum cost. We show that this problem can be solved in $O(mn_1)$ time, where m is the number of edges and n_1 is the number of people. Manlove and Sng [9] showed an $O((\sqrt{C} + n_1)m)$ algorithm for this problem without costs, where C is the sum of capacities of all items. They called it Capacitated House Allocation with Ties (CHAT) and the problem was to determine if G admits a popular matching or not, and if so, to compute one. Thus our algorithm improves upon the algorithm in [9].

1.2 Background

Abraham et al. [2] considered the problem of determining if a given graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ admits a popular matching or not, and if so, computing one. They also gave a structural characterization of graphs that admit popular matchings. Section 2 outlines this characterization and the algorithm that follows from it.

Subsequent to this work, there have been several variants of the popular matchings problem considered. One line of research has been on generalizations of the popular matchings problem while the other direction has been to deal with instances that do not admit any popular matchings. The generalizations include the capacitated version studied by Manlove and Sng [9], the weighted version studied by Mestre [12] and random popular matchings considered by Mahdian [8]. Kavitha and Nasre [6] as well as McDermind and Irving [11] independently studied the problem of computing an *optimal* popular matching for strict instances where the notion of optimality is specified as a part of the input. Note that they also consider the min-cost popular matchings but in that case the costs are associated with edges whereas in our case costs are associated with items. As described earlier, the line of research pursued for instances that do not admit popular matchings includes the NP-hardness of least unpopular matchings [10], the existence and algorithms for popular mixed matchings [5] and NP-hardness of the popular matchings problem with variable job capacities [7].

Organization of the paper. Section 2 discusses preliminaries. Section 3 shows that the min-cost popular instance problem is NP-hard. Section 4 has our results for the min-cost augmentation problem and Section 5 has our algorithm for the min-cost popular matching problem.

2 Preliminaries

We review the characterization of popular matchings given in [2]. Let $G_1 = (\mathcal{A} \cup \mathcal{B}, E_1)$ be the graph containing only rank-1 edges. Then [2, Lemma 3.1] shows that a matching M is popular in G only if

$M \cap E_1$ is a maximum matching of G_1. Maximum matchings have the following important properties, which we use throughout the rest of the paper.

$M \cap E_1$ defines a partition of $\mathcal{A} \cup \mathcal{B}$ into three disjoint sets: a vertex $u \in \mathcal{A} \cup \mathcal{B}$ is *even* (resp. *odd*) if there is an even (resp. odd) length alternating path in G_1 (w.r.t. $M \cap E_1$) from an unmatched vertex to u. Similarly, a vertex u is *unreachable* if there is no alternating path from an unmatched vertex to u. Denote by $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ the sets of even, odd, and unreachable vertices, respectively.

Lemma 1 (Gallai-Edmonds Decomposition). *Let $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ be the sets of vertices defined by G_1 and $M \cap E_1$ above. Then*

(a) $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ are pairwise disjoint, and independent of the maximum matching $M \cap E_1$.
(b) In any maximum matching of G_1, every vertex in $\mathcal{O}$ is matched with a vertex in $\mathcal{E}$, and every vertex in $\mathcal{U}$ is matched with another vertex in $\mathcal{U}$. The size of a maximum matching is $|\mathcal{O}| + |\mathcal{U}|/2$.
(c) No maximum matching of G_1 contains an edge between a vertex in $\mathcal{O}$ and a vertex in $\mathcal{O} \cup \mathcal{U}$. Also, G_1 contains no edge between a vertex in $\mathcal{E}$ and a vertex in $\mathcal{E} \cup \mathcal{U}$.

Since every maximum cardinality matching in G_1 matches all vertices $u \in \mathcal{O} \cup \mathcal{U}$, these vertices are called *critical* as opposed to vertices $u \in \mathcal{E}$ which are called *non-critical*. Using this partition of vertices, the following definitions can be made.

Definition 2. *For each $a \in \mathcal{A}$, define $f(a)$ to be the set of top choice items for a. Define $s(a)$ to be the set of a's most-preferred* non-critical *items in G_1.*

Theorem 1 (from [2]). *A matching M is popular in G iff (i) $M \cap E_1$ is a maximum matching of $G_1 = (\mathcal{A} \cup \mathcal{B}, E_1)$, and (ii) for each applicant a, $M(a) \in f(a) \cup s(a)$.*

The algorithm for solving the popular matching problem is now straightforward: each $a \in \mathcal{A}$ determines the sets $f(a)$ and $s(a)$. A matching that is maximum in G_1 and that matches each a to an item in $f(a) \cup s(a)$ needs to be determined. If no such matching exists, then G does not admit a popular matching.

3 Min-cost Popular Instance

In this section we consider the Min-Cost Popular Instance problem. Our input is a set $\mathcal{A}$ of people where each $a \in \mathcal{A}$ has a preference list over items in a universe U, where each item $b \in U$ has a price $\mathsf{cost}(b) \geq 0$. The problem is to "construct" a graph G or equivalently, set suitable values of $\mathsf{cap}(b)$ for each $b \in U$, in order to ensure that the resulting graph G admits a popular matching that matches all $a \in \mathcal{A}$, at the least possible cost.

We will show that the above problem is NP-hard by showing a reduction from the monotone 1-in-3 SAT problem to this problem. The monotone 1-in-3 SAT problem is a variant of the 3SAT problem where each clause contains exactly 3 literals and no literal appears in negated form. The monotone 1-in-3 SAT problem asks if there exists a satisfying assignment to the variables such that each clause has exactly 1 literal set to true. This problem is NP-hard [13].

Let $\mathcal{I}$ be an instance of the monotone 1-in-3 SAT problem. Let $C_1, \ldots, C_m$ be the clauses in $\mathcal{I}$ and let $X_1, \ldots, X_n$ be the variables in $\mathcal{I}$. We construct from $\mathcal{I}$ an instance of the min-cost popular instance problem as follows:

Corresponding to each clause $C_i = (X_{j_1} \vee X_{j_2} \vee X_{j_3})$, we have 9 people $A_i = \{a_1^i \ldots, a_9^i\}$. Their preference lists are shown in Fig. 1. In this case every person has a preference list of length 2, that is a top item followed by a second choice item. For instance, a_1^i treats item u_{j_1} as its rank-1 item and item u_{j_2} as its rank-2 item.

The items $u_{j_1}, u_{j_2}, u_{j_3}$ are called *public* items and the items p_1^i, p_2^i, p_3^i, and q^i are called *internal* items. The internal items appear only on the preference lists of the people of A_i while the public item u_j corresponds to the variable X_j. In every clause C_i that X_j belongs to, the item u_j appears in the preference lists of some of the people in the set A_i as shown above in Fig. 1.

The set $\mathcal{A}$ of people in our instance is $\cup_i A_i$. The universe U of all items is union of $\{u_1, \ldots, u_n\}$ (the n public items) and the set $\cup_i \{p_1^i, p_2^i, p_3^i, q^i\}$ of all the internal items. It remains to describe the costs of

a_1^i	u_{j_1}	u_{j_2}	
a_2^i	u_{j_2}	u_{j_3}	
a_3^i	u_{j_1}	u_{j_3}	

a_4^i	u_{j_1}	p_1^i	
a_5^i	u_{j_2}	p_2^i	
a_6^i	u_{j_3}	p_3^i	

a_7^i	p_1^i	q^i	
a_8^i	p_2^i	q^i	
a_9^i	p_3^i	q^i	

Fig. 1. The preference lists of people corresponding to the i-th clause in $\mathcal{I}$.

the items. For each i, the cost of each p_t^i for $t = 1, 2, 3$, is 1 unit, while the cost of q^i is zero units. The cost of each u_j, for $j = 1, \ldots, n$, is 3 units.

Recall that our problem is to determine a set $\mathcal{B}$ of items with suitable capacities so that the graph $(\mathcal{A} \cup \mathcal{B}, E)$ admits a popular matching that matches all $a \in \mathcal{A}$ and we want to do this at the least possible cost. We first show the following lemma.

Lemma 2. *Any instance $(\mathcal{A} \cup \mathcal{B}, E)$ that admits a popular matching that matches all $a \in \mathcal{A}$ has cost at least $14m$.*

Proof. Let us focus on the set A_i of people corresponding to clause C_i. The preference lists of people in A_i are shown in Fig. 1. Since the cost of each item on the lists of a_1^i, a_2^i, a_3^i is 3, we have to spend 9 units to buy an item each for these 3 people (since we seek an instance where all the persons get matched). People a_4^i, a_5^i, a_6^i have a unit cost item in their preference lists (items p_1^i, p_2^i, p_3^i, respectively). Thus we have to spend 3 units to buy an item each for these 3 people. Finally, a_7^i, a_8^i, a_9^i have a cost 0 item, i.e. q^i, in their preference lists. Hence we can get q^i with $\mathsf{cap}(q^i) = 3$ for a cost of 0. Summarizing, we need to spend at least $9 + 3 + 0 = 12$ units for the people in A_i.

However it is not possible to spend just 12 units for the people in A_i. This is because in the first place, we are forced to have positive capacities for at least 2 of the 3 items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ and if we seek to match a_4^i to p_1^i while u_{j_1} is around, then p_1^i is a_4^i's second choice item. Since p_1^i is a_7^i's top choice item, we also have to match a_7^i to p_1^i since a popular matching has to be a maximum cardinality matching on rank-1 edges (see Theorem 1). Thus it is not possible to match a_7^i to q^i in a popular matching while p_1^i gets matched to a_4^i who regards this item as a second choice item because u_{j_1} is around.[5] Thus we have the following options:

(i) set $\mathsf{cap}(u_{j_1}) = 0$ and then match a_4^i to p_1^i and a_7^i to q^i
(ii) match both a_4^i and a_7^i to p_1^i by setting $\mathsf{cap}(p_1^i) = 2$
(iii) increase the capacity of u_{j_1} by 1 more and set $\mathsf{cap}(p_1^i) = 0$ and thus match a_4^i to u_{j_1} and a_7^i to q^i

It is not possible to have option (i) for all the u_j's since we are forced to have positive capacities for at least 2 of the 3 items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$. Note that (ii) is always better than (iii) (cost of 2 units vs cost of 3 units). Hence it is always cheaper to match a_4^i, a_5^i, a_6^i to p_1^i, p_2^i, p_3^i respectively than to any of $u_{j_1}, u_{j_2}, u_{j_3}$.

Thus when we match a_4^i, a_5^i, a_6^i to p_1^i, p_2^i, p_3^i respectively, at least 2 of these 3 people are getting matched to their second choice items. Hence at least 2 out of the 3 people among a_7^i, a_8^i, a_9^i will also have to be matched to their top choice items in order to ensure that the resulting matching is popular. This implies a cost of at least $9 + 3 + 2 = 14$ for A_i.

This holds for each A_i, where $1 \leq i \leq m$. Since the cost is at least 14 per clause, it amounts to at least $14m$ in total for all the clauses. □

The following lemma establishes the correspondence between the instance $\mathcal{I}$ of monotone 1-in-3-SAT and the min-cost popular instance that we defined.

Lemma 3. *There exists an instance $(\mathcal{A} \cup \mathcal{B}, E)$ with cost $14m$ that admits a popular matching that matches all $a \in \mathcal{A}$ iff there exist a 1-in-3 satisfying assignment for $\mathcal{I}$.*

[5] A matching that contains the 3 edges $(a_1^i, u_{j_1}^i), (a_4^i, p_1^i), (a_7^i, q^i)$ cannot be popular since by promoting a_7^i from q^i to p_1^i, and a_4^i from p_1^i to u_{j_1}, and leaving a_1^i unmatched, we get a more popular matching.

Proof. We know from Lemma 2 that any instance $(\mathcal{A} \cup \mathcal{B}, E)$ that admits a popular matching that matches all $a \in \mathcal{A}$ has a cost of at least $14m$. What we need to show here is that $(\mathcal{A} \cup \mathcal{B}, E)$ has cost $14m$ if and only if the 1-in-3-SAT instance $\mathcal{I}$ is a "yes" instance, that is, there is a true/false assignment to the variables $X_1, \ldots, X_n$ such that each clause has exactly 1 literal set to true (and thus 2 literals set to false).

Suppose $\mathcal{I}$ admits such an assignment. We now show how to construct a set $\mathcal{B}$ of cost $14m$ such that the instance $(\mathcal{A} \cup \mathcal{B}, E)$ admits a popular matching that matches all $a \in \mathcal{A}$. If $X_i = \mathit{true}$ then set $\mathsf{cap}(u_i) = 0$, else $\mathsf{cap}(u_i)$ will be set to a suitable strictly positive value.

Since the setting of true/false values to X_i's is a satisfying assignment, every clause has two literals set to false and 1 set to true. Let clause C_i be $(X_{j_1} \vee X_{j_2} \vee X_{j_3})$. Thus there is 1 variable X_{j_k} in $\{X_{j_1}, X_{j_2}, X_{j_3}\}$ that has been set to true. By our definition of capacities, the corresponding u_{j_k} has 0 capacity. Hence the people in the set A_i can be matched as follows:

- a_1^i, a_2^i, a_3^i get matched to the 2 items in $\{u_{j_1}, u_{j_2}, u_{j_3}\} \setminus \{u_{j_k}\}$ by having 2 copies of one of the lower indexed item and 1 copy of the higher indexed item for these 3 people.
- p_k^i becomes a_{k+3}^i's top choice item (since u_{j_k} does not exist in the graph now) and hence we can now match a_{k+3}^i to p_k^i and a_{k+6}^i to q^i.

This way we spend only $9 + 3 + 2 = 14$ units for the people in A_i and each person a has an item in $f(a) \cup s(a)$ to be matched to. Since every clause in $\mathcal{I}$ has 1 exactly variable set to true and 2 set to false, we achieve a cost of 14 for each set A_i. This shows that we can construct a set $\mathcal{B}$ of cost $14m$ such that $(\mathcal{A} \cup \mathcal{B}, E)$ admits a popular matching that matches all $a \in \mathcal{A}$.

To show the other direction, let us set the true/false values of variables in $\mathcal{I}$ as follows: for each $j = 1, \ldots, n$ set $X_j = \mathit{true}$ if and only if $\mathsf{cap}(u_j) = 0$. We need to show that such an assignment sets exactly 1 variable in each clause to be true.

Let us consider any clause $C_i = (X_{j_1} \vee X_{j_2} \vee X_{j_3})$. Among the 3 items $u_{j_1}, u_{j_2}, u_{j_3}$ that correspond to these 3 variables we need capacities of at least 2 items to be positive so as to match all the 3 people a_1^i, a_2^i, a_3^i. Thus our true/false assignment does not set more than 1 variable per clause to true.

We now need to show that there is at least 1 item in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ with capacity 0. This is where we will use the hypothesis that we can construct $(\mathcal{A} \cup \mathcal{B}, E)$ of cost $14m$ that admits a popular matching that matches all $a \in \mathcal{A}$. It follows from the proof of Lemma 2 that each set A_i of people corresponding to a clause needs a cost of at least 14. Since the total cost is only $14m$ and there are m clauses, this implies that we have to spend exactly 14 per clause. In other words, the items for the 9 people of each A_i have to be bought using only 14 units.

If all the 3 items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ have positive capacities, then this implies the cost of items for all the 9 people in A_i will be $9 + 3 + 3 = 15$ since when each u_{j_k} has positive capacity, then the u_{j_k}'s become top choice items for a_4^i, a_5^i, a_6^i, respectively and thus p_1^i, p_2^i, p_3^i become their second choice items. This forces us to match each of a_7^i, a_8^i, a_9^i to their top choice items (that is, p_1^i, p_2^i, p_3^i, respectively) since a popular matching has to be a maximum cardinality matching on rank-1 edges. However we are given that we can spend only 14 units per A_i; thus it has to be the case that there exists at least 1 item in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ with capacity 0. This finishes the proof of this lemma. □

Note that the preference lists of all the people in our instance G are strict and of length at most 2. Also, the preference lists are drawn from a *master list*. We have thus shown the following theorem.

Theorem 2. *The min-cost popular instance problem is NP-hard, even when each preference list has length at most 2. Further, the hardness holds even when the preference lists are derived from a master list.*

4 Min-cost Augmentation

In this section we show various results for the min-cost augmentation problem. Recall that the input here is a graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ where each item $b \in \mathcal{B}$ has a non-negative $\mathsf{cost}(b)$ associated with it. The problem is to determine how to increase the capacities of items in $\mathcal{B}$ so that the resulting graph admits a popular matching and the cost of the increase in capacities is minimized.

Unlike the min-cost popular instance problem, the above problem admits a simple polynomial time algorithm when each $a \in \mathcal{A}$ has a preference list that is strict and of length at most 2. We describe this algorithm below. We assume throughout this section that we add at the end of each a's preference list a dummy item called the *last item* ℓ_a, where a being matched to ℓ_a amounts to a being left unmatched.

4.1 Preference lists of length 2

For any $a \in \mathcal{A}$, a's preference list consists of a top choice item (let us use f_a to denote this item), and possibly a second choice item (let us use z_a to denote this item) and then of course, the last item ℓ_a that we added for convenience. Let G_1 be the graph G restricted to rank-1 edges. Let the graph $G' = (\mathcal{A} \cup \mathcal{B}, E')$, where E' consists of

- all the top ranked edges (a, f_a): one such edge for each $a \in \mathcal{A}$, and
- the edges (a, s_a), where a is even in G_1 and s_a is a's most preferred item that is even in G_1. Thus $s_a = z_a$ when z_a is nobody's top choice item, else $s_a = \ell_a$.

It follows from Theorem 1 that G admits a popular matching if and only if G' admits an $\mathcal{A}$-perfect matching. We assume that G does not admit a popular matching and we have to decide now the capacities of which items should get increased and by how much.

If G' does not admit an $\mathcal{A}$-perfect matching, then consider the set S of people whose neighborhood $N(S)$ in G' has size smaller than $|S|$. Every $a \in S$ must satisfy $s_a = z_a$. Otherwise, $s_a = \ell_a$ and since no vertex in $\mathcal{A}$ other than a, has an edge to ℓ_a, such an a will always be matched in any maximum cardinality matching in G' and hence cannot belong to S.

Since $s_a = z_a$ for every $a \in S$ and every person's preference list has length 2, there are no items sandwiched between f_a and s_a in a's preference list for every $a \in S$. Hence it is only the items in the neighborhood $N(S)$ of S in G' whose capacities need to be increased so that the new G' admits an $\mathcal{A}$-perfect matching.

Our algorithm to increase item capacities is the following:

- Let M be a maximum cardinality matching in G'.
- While M is not $\mathcal{A}$-complete do
  - Identify all the items that are *odd* with respect to M; call this set of items $\mathcal{O}'$.
  - Let b be a cheapest item in $\mathcal{O}'$. Set $\mathsf{cap}(b) = \mathsf{cap}(b) + 1$.
  - Augment M along the new augmenting path now available.

Our increase of capacities ensures that no top choice item f_a in $N(S)$ ever becomes *even* in G_1. First, every a that is unmatched in M has to be even in G_1. The set $\mathcal{O}'$ is identified by constructing alternating paths from unmatched vertices with respect to M. No item u that was unreachable in G_1 ever occurs in any such alternating path since every $\alpha \in \mathcal{A}$ on this path has to be even in G_1 and u cannot be f_α or s_α for any such α. Regarding items that are odd in G_1, such items b do occur in these alternating paths, however $\mathsf{cap}(b)$ will always be bounded by b's degree in G', which is the same as b's degree in G_1. Since preference lists are strict, b with capacity $\mathsf{cap}(b)$ remains critical in G_1.

Thus our increase of capacities of items in $N(S)$ will ensure that an item that is *critical* in G_1 will continue to remain critical in G_1. So the most preferred even item in G_1 will not change for any $a \in \mathcal{A}$ because of our increase of item capacities.

Let $\tilde{G}$ be the graph that corresponds to the increased item capacities as given by the algorithm above. By our increase of item capacities, it is obvious that new $\tilde{G}'$ admits an $\mathcal{A}$-complete matching or in other words, $\tilde{G}$ admits a popular matching. It follows from our algorithm that there is no alternating path between any item b whose capacity was increased by our algorithm and an item β whose cost is strictly smaller than $\mathsf{cost}(b)$. Hence we can conclude the following lemma.

Lemma 4. *$\tilde{G}$ is the min-cost augmentation of G that admits a popular matching.*

It is easy to see that the running time of the algorithm is $O(n_1^2)$, where n_1 is the size of $\mathcal{A}$. Hence we have shown the following theorem.

Theorem 3. *The min-cost augmentation problem with strict preference lists of length at most 2 can be solved in $O(n_1^2)$ time.*

4.2 Hardness for the general case

We now show that the min-cost augmentation problem in the general case is NP-hard. The reduction is again from the monotone 1-in-3 SAT problem (refer to Section 3). Let $\mathcal{I}$ be an instance of the monotone 1-in-3 SAT problem. Let $C_1, \ldots, C_m$ be the clauses in $\mathcal{I}$ and let $X_1, \ldots, X_n$ be the variables in $\mathcal{I}$. We construct from $\mathcal{I}$ an instance of the min-cost augmentation problem as follows.

Let C_i be $(X_{j_1} \vee X_{j_2} \vee X_{j_3})$. Corresponding to this clause we have 6 people $A_i = \{a_1^i, a_2^i, a_3^i, a_4^i, a_5^i, a_6^i\}$ and 3 internal items $D_i = \{p_i, q_i, r_i\}$. In addition we have public items $u_{j_1}, u_{j_2}, u_{j_3}$ which belong to preference lists of people in A_i and whenever X_j occurs in a clause C_i, the item u_j will belong to the preference lists of some people in A_i. The public items have unit cost whereas each internal item $b \in D_i$ has cost 2. The preference lists of the people in A_i are shown in Fig. 2.

a_1^i	p_i	u_{j_1}	q_i
a_2^i	p_i	u_{j_2}	q_i
a_3^i	p_i	u_{j_3}	q_i

a_4^i	r_i	u_{j_1}
a_5^i	r_i	u_{j_2}
a_6^i	r_i	u_{j_3}

Fig. 2. Preference lists of the 6 people in A_i

The set $\mathcal{B}$ of items is the union of $\cup_{i=1}^{m} D_i$ (the set of all the internal items) and $\{u_1, \ldots, u_n\}$ (consisting of all the public items, where vertex u_j corresponds to the j-th variable X_j). The set $\mathcal{A}$ of people is the union of $\cup_{i=1}^{m} A_i$ and $\{x_1, \ldots, x_n\}$, where the vertex x_j corresponds to the variable X_j. The preference list of each x_j is of length 1, it consists of the item u_j.

G has no popular matching. It is easy to see that the graph G described above does not admit any popular matching. To see this, first note that each public item u_j is a unique rank-1 item for exactly one applicant x_j. Hence when the capacity of every item is 1, these public items are unreachable or critical in G_1 (the subgraph of rank-1 edges in G). Now let us consider the people in A_i: for each $a_t^i \in \{a_1^i, a_2^i, a_3^i\}$, we have $f(a_t^i) = \{p_i\}$ and $s(a_t^i) = \{q_i\}$. Since there are only 2 items p_i, q_i for the 3 people a_1^i, a_2^i, a_3^i to be matched to in any popular matching, G does not admit a popular matching.

Let $\tilde{G}$ be a min-cost instance augmentation such that $\tilde{G}$ admits a popular matching. We now state the following lemma that establishes the reduction.

Lemma 5. *$\tilde{G}$ has cost at most m iff there exists a 1-in-3 satisfying assignment for the instance $\mathcal{I}$.*

Proof. Assume that there exists a 1-in-3 satisfying assignment for $\mathcal{I}$. For each j, let c_j denote the number of clauses in which X_j appears. We will set the capacities of the items in the following manner: the capacities of the internal items remain the same, i.e., $\mathsf{cap}(b) = 1$ for each $b \in \cup_i D_i$ and the capacities of the public items are set as follows.

For each j, where $1 \leq j \leq n$ do:

- if $X_j = true$, then set $\mathsf{cap}(u_j) = 1 + c_j$
- else $\mathsf{cap}(u_j)$ remains 1.

Let us determine the cost of this augmentation. For every X_j that is true, we pay a cost of $c_j \cdot 1 = c_j$ and for X_j that is false, we pay nothing. Since each clause has exactly one variable set to true, we have: $\sum_{j:X_j=true} c_j = m$. Thus the cost of our augmentation is m.

We now show that the graph $\tilde{G}'$ admits an $\mathcal{A}$-perfect matching (the edges in $\tilde{G}'$ are (a, b) where $b \in f(a) \cup s(a)$).

- Consider the people $x_1, \ldots, x_n$. Each x_j gets matched to her f-item u_j.
- Consider the people in A_i. We know that exactly one amongst $u_{j_1}, u_{j_2}, u_{j_3}$ has its capacity greater than 1 (since our assignment of capacities was based on a satisfying assignment for 1-in-3 SAT). If $\mathsf{cap}(u_{j_k}) > 1$, then a_k^i gets matched to u_{j_k} and the 2 people in $\{a_1^i, a_2^i, a_3^i\} \setminus \{a_k^i\}$ get matched to p_i and q_i. Finally, a_{k+3}^i gets matched to her top choice item r_i whereas the 2 people in $\{a_4^i, a_5^i, a_6^i\} \setminus \{a_{k+3}^i\}$ get matched to their last items (their most preferred even item in G_1).

To prove the other direction, assume that the cost of $\tilde{G}$ is m. We now translate this into truth values for variables in $\mathcal{I}$. If $\mathsf{cap}(u_j) > 1$ in $\tilde{G}$, then set variable $X_j = true$, else set $X_j = false$. We need to show that this is a 1-in-3 satisfying assignment for $\mathcal{I}$.

Since the cost of increasing the capacity of any item is at least 1, we need to pay at least 1 unit per clause in order to match the people in A_i. Thus we need to pay at least m to get a graph that admits a popular matching. However we are given that with a cost of exactly m, the graph $\tilde{G}$ that admits a popular matching. Hence the capacities of items have been augmented such that exactly 1 unit has been spent per clause.

Spending 1 unit has allowed all the people in A_i, for each i, to have enough items to match themselves to in $\tilde{G}'$. Consider the items that occur in the preference lists of people in A_i (refer to Fig. 2). Since the cost of each internal item is 2 and we cannot afford a cost of 2 for any clause, it has to be the case that $\mathsf{cap}(u) > 1$ for some $u \in \{u_{j_1}, u_{j_2}, u_{j_3}\}$. Thus we have at least 1 true variable per clause in $\mathcal{I}$.

We now have to show that there is exactly 1 true variable per clause in $\mathcal{I}$. The point to note is that $\mathsf{cap}(u) > 1$ for any public item u implies that u is non-critical in $\tilde{G}_1$. This changes the *most preferred even* item in $\tilde{G}_1$ for some people. That is, suppose k items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ have their capacities larger than 1. Then we have k non-critical items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ and so we have k people in $\{a_4^i, a_5^i, a_6^i\}$ satisfying the following: a's *most preferred even* item in $\tilde{G}_1$ is no longer the last resort item ℓ_a, it is now the non-critical public item that is second in a's preference list.

Observe that one person in $\{a_4^i, a_5^i, a_6^i\}$ can be matched to her top choice item r_i. However, to match the second person we need to spend another unit. In the first place, we have already spent 1 unit to increase some $\mathsf{cap}(u_{j_k})$ to match all the people in $\{a_1^i, a_2^i, a_3^i\}$. With more than one item in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ non-critical in $\tilde{G}_1$, we have pay at least 2 units for the people in A_i. This contradicts the fact that we spent exactly 1 unit for the people in A_i. Hence there is exactly 1 true variable per clause in $\mathcal{I}$. □

We can now conclude the following theorem.

Theorem 4. *The min-cost augmentation problem is NP-hard, even for strict lists of length at most 3. Further, the lists can be derived from a master list.*

4.3 Inapproximability of min-cost augmentation

We can extend the above reduction from $\mathcal{I}$ to show that this problem is NP-hard to approximate to within a factor of $\sqrt{n_1}/2$, where n_1 is the size of $\mathcal{A}$. We construct a graph H on at most $4m^4$ people that satisfies the following property:

(∗) If $\mathcal{I}$ is a *yes* instance for 1-in-3 SAT, then H can be augmented at a cost of m to admit a popular matching. If $\mathcal{I}$ is a *no* instance for 1-in-3 SAT, then H needs a cost strictly greater than m^3 to admit a popular matching.

We describe the construction of the graph H below. Recall that $\mathcal{I}$ has m clauses. Corresponding to each clause C_i, we have a set A_i of people. The construction of H is as follows. Let us call the group of 3 people (a_4^i, a_5^i, a_6^i) in Fig. 2 a *triplet*. Instead of having just one triplet in A_i as was the case in the previous section, here we have many such triplets. In particular, we have $m^3 + 1$ such triplets. The preference list for one particular triplet $(a_{3t+1}^i, a_{3t+2}^i, a_{3t+3}^i)$ is shown in Fig. 3.

a_{3t+1}^i	r_t^i	u_{j_1}	
a_{3t+2}^i	r_t^i	u_{j_2}	
a_{3t+3}^i	r_t^i	u_{j_3}	

Fig. 3. Preference lists of people corresponding to the t-th triplet

We now have $3 + 3(m^3 + 1)$ people in A_i, namely a_1^i, a_2^i, a_3^i and 3 people per triplet, for each of the $m^3 + 1$ triplets. Thus our overall instance H has $m(3 + 3(m^3 + 1))$ (the people in $\cup_i A_i$), plus the n

people in $\{x_1, \ldots, x_n\}$. Since each clause has 3 variables, $n \leq 3m$. Thus we can bound n_1, the number of applicants in H as: $n_1 \leq 3m^4 + 9m \leq 4m^4$ for $m \geq 3$.

Recall that for each i, the preference list of x_j is of length 1, which consists of only u_j. The costs of the items are as follows: the cost of each of the *internal* items, i.e., p_i, q_i, and r_t^i, for $t = 1, \ldots, m^3 + 1$ is m^3, and the cost of each u_j for $j = 1, \ldots, n$ is 1. The proof of $(*)$ is given in the Appendix.

Suppose the min-cost augmentation problem admits a $\sqrt{n_1}/2$ approximation algorithm. Call this algorithm Algo1. If $\mathcal{I}$ is a yes instance, then Algo1 has to return an augmentation of cost at most $1/2.\sqrt{4m^4}.m = m^3$. If $\mathcal{I}$ is a no instance, then there is no augmentation of cost at most m^3, so Algo1 returns an answer of cost greater than m^3. Thus Algo1 determines whether $\mathcal{I}$ has a 1-in-3 satisfying assignment or not. Hence we can conclude the following theorem.

Theorem 5. *It is NP-hard to approximate the min-cost augmentation problem on $G = (\mathcal{A} \cup \mathcal{B}, E)$ to within $\sqrt{|\mathcal{A}|}/2$.*

5 Min-cost Popular matchings

In this section we present an $O(mn_1)$ time algorithm for the min-cost popular matchings problem. Our input is an instance $G = (\mathcal{A} \cup \mathcal{B}, E)$ where each item $b \in \mathcal{B}$ has a capacity $\mathsf{cap}(b)$ (denoting the maximum number of people that can be matched to b) and a price $\mathsf{cost}(b) \geq 0$. Whenever a person gets matched to b, an amount of $\mathsf{cost}(b)$ has to be paid. Thus if a capacity of $k \leq \mathsf{cap}(b)$ of b gets used in a matching M, then a cost of $k \cdot \mathsf{cost}(b)$ has to be paid by M. As done in the earlier sections, we will add a last item ℓ_a at the end of a's preference list for each person $a \in \mathcal{A}$. The cost of ℓ_a is 0, since using the edge (a, ℓ_a) amounts to leaving a unmatched.

Our problem here is to decide whether G admits a popular matching or not and if so, to compute the one with minimum cost. As mentioned in Section 1, Manlove and Sng considered the popular matchings problem (referred to as the CHAT problem) where items (these were called houses) have capacities and they showed an $O((\sqrt{C} + n_1)m)$ algorithm for this problem, where $m = |E|$, $n_1 = |\mathcal{A}|$, and C is the sum of all the capacities.

In order to solve the min-cost popular matchings problem, for each $b \in \mathcal{B}$, we could make $\mathsf{cap}(b)$ copies of each vertex b, call them $b_1, \ldots, b_{\mathsf{cap}(b)}$, where each b_i has the same neighborhood as the original vertex b. However, such a graph has too many vertices and edges, hence we will stick to the original graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ and simulate the larger graph in G itself. Note that a *matching* in G can contain up to $\mathsf{cap}(b)$ pairs (a_i, b). It is easy to see that the structural characterization for popular matchings from [2] holds for our problem as well. That is, any popular matching in our graph G has to be a maximum cardinality matching on rank-1 edges and every person a has to be matched to an item in $f(a) \cup s(a)$. This is because by making $\mathsf{cap}(b)$ many copies of every item b in G our problem becomes equivalent to the original popular matchings problem.

5.1 Our algorithm

We first construct the graph G_1 which is the graph G restricted to rank-1 edges. In order to find a maximum cardinality matching in the graph G_1, we use Ford-Fulkerson max-flow algorithm. The following transformation from G_1 into a flow network is based on the standard transformation from the bipartite matching problem to the maximum flow problem:

- add a vertex s and an edge directed from s to each person $a \in \mathcal{A}$ with an edge capacity of 1 on this edge.
- add a vertex t and an edge directed from each item $b \in \mathcal{B}$ to t with an edge capacity of $\mathsf{cap}(b)$ on this edge.
- direct every edge (a, b) of G from a to b and set an edge capacity of 1 for each such edge.

Let $F(G_1)$ denote the above graph. It is easy to see that a valid flow from s to t in the graph $F(G_1)$ can be translated to a *matching* in G_1 in which every applicant is matched to at most 1 item and every item b is matched up to $\mathsf{cap}(b)$ people. A maximum flow in $F(G_1)$ becomes a maximum cardinality matching in G_1. We now describe our algorithm which can be broadly partitioned into 2 stages. The first stage builds the graph G' whereas the second stage computes a min-cost popular matching if one exists.

The first stage. Compute a maximum cardinality matching M_0 of G_1 by computing a max-flow from s to t in $F(G_1)$. Using the matching M_0, obtain a partition of $\mathcal{A} \cup \mathcal{B}$ into $\mathcal{O}$ (*odd*), $\mathcal{E}$ (*even*) and $\mathcal{U}$ (*unreachable*). We note that since we do not make $\mathsf{cap}(b)$ many copies of every item b, it is not obvious how to obtain such a partition. We give the details for this step in the Appendix. Assuming that we have obtained a partition, for each $a \in \mathcal{A}$, let $s(a)$ be the set of a's most preferred items in $\mathcal{E}$. Let the graph G' be the graph G_1 along with the edges (a, b) where $a \in \mathcal{E}$ and $b \in s(a)$.

Since a popular matching is a maximum cardinality matching on rank-1 edges, every item that is *critical* in G_1, that is, all items in $\mathcal{O} \cup \mathcal{U}$ have to be fully matched in every popular matching M^* of G. The only choice we have is in choosing which items of $\mathcal{E}$ should participate to what capacity in the min-cost popular matching. We make this choice in the second stage of our algorithm.

The second stage. The second part of our algorithm is to augment the matching M_0 to find a min-cost popular matching. Recall that it is items that have costs. Let ρ be an augmenting path with respect to M_0, i.e., one end of ρ is an unmatched person and the other end of ρ is an item β that is not fully matched. The cost of augmenting the current matching along ρ is the cost of β. By augmenting the current matching along ρ, every item that is currently matched stays matched to the same number of people and the item β gets matched to one more person. Thus the cost of the new matching is the cost of the old matching + $\mathsf{cost}(\beta)$. In order to match an unmatched person a, our algorithm always chooses the cheapest augmenting path starting from the person a.

The main points to note in the second stage of our algorithm are the following:

- Our starting matching is M, where $M = M_0 \setminus \{\text{all edges } (a, b) \text{ where } a \in \mathcal{O}\}$. Thus M consists only of edges (a, b) where $b \in \mathcal{O} \cup \mathcal{U}$. We take M to be our starting matching rather than M_0 because it may be possible to match vertices in $\mathcal{O} \cap \mathcal{A}$ to cheaper rank-1 neighbors. Recall that while computing the max-flow M_0, the costs of items played no role.
- For every person a that is unmatched in M_1, we build a Hungarian tree T_a rooted at a. Initially all vertices are unmarked and while building T_a, every visited vertex get marked so that each vertex occurs at most once in T_a.
- If T_a has no augmenting path, then quit and return "G admits no popular matching".
- The construction of the tree T_a is not terminated as soon as we find an augmenting path but we build T_a completely in order to find a min-cost item β such that there is an augmenting path between a and β.

It is easy to see that if there is no augmenting path in T_a, where a is an unmatched person in M, then there is no popular matching in G. This is because every popular matching is a maximum cardinality matching on rank-1 edges and has to match every $a \in \mathcal{A}$ to an item in $f(a) \cup s(a)$. It remains to prove that if G admits a popular matching, then our algorithm always finds a min-cost popular matching.

Lemma 6. *If G admits a popular matching, then the matching M returned by our algorithm is a min-cost popular matching in G.*

Proof. Suppose M is not a min-cost popular matching in G and let OPT be such a matching. For the purpose of this proof we operate on the *cloned* graph where each item b has $\mathsf{cap}(b)$ many copies and M and OPT both refer to matchings where each item is matched to at most one person. Consider $\mathsf{OPT} \oplus M$ - this is a collection of cycles and even length alternating paths (since both OPT and M are $\mathcal{A}$-complete). The cycles do not contribute to any change in costs since both OPT and M match the same items in any cycle.

Let ρ be a path in $\mathsf{OPT} \oplus M$. Let β_0 and β_M be the endpoints of this path, where OPT leaves β_M unmatched while M leaves β_0 unmatched. It suffices to show that $\mathsf{cost}(\beta_M) \leq \mathsf{cost}(\beta_0)$. Since OPT is a popular matching, it has to match all the items in $\mathcal{O} \cup \mathcal{U}$ (the odd/unreachable items in G_1). Since it leaves β_M unmatched, it follows that $\beta_M \in \mathcal{E}$ and thus there are items of $\mathcal{E}$ in ρ.

It is the second stage of our algorithm that matches items in $\mathcal{E}$. Let α_1 be the last person in the path ρ to get matched by our algorithm and let $M(\alpha_1) = \beta_1$. Since β_0 is unmatched in M it implies that during the execution of our algorithm we found at least two augmenting paths from α_1 – one ending in β_1 and the other ending in β_0. Further, we found the augmenting path ending in β_1 cheaper, that is, $\mathsf{cost}(\beta_1) \leq \mathsf{cost}(\beta_0)$.

We now repeat the same argument for the β_1-β_M subpath of ρ. Let α_2 be the last person in the β_1-β_M subpath that got matched by our algorithm and let $M(\alpha_2) = \beta_2$. Note that β_1 was also unmatched at this time and hence our algorithm found at least two augmenting paths from α_2 – one ending in β_1 and another ending in β_2. Since $M(\alpha_2) = \beta_2$ it implies that $\mathsf{cost}(\beta_2) \leq \mathsf{cost}(\beta_1)$.

Repeating the same argument for the β_2-β_M subpath we get vertices $\beta_3, \ldots, \beta_t = \beta_M$ where $\mathsf{cost}(\beta_2) \leq \mathsf{cost}(\beta_3) \leq \cdots \leq \mathsf{cost}(\beta_t)$. Combining all the inequalities yields $\mathsf{cost}(\beta_M) \leq \mathsf{cost}(\beta_0)$. □

Time complexity of this algorithm. The difference between our algorithm and that of Manlove and Sng for the CHAT problem in the first stage is that they use Gabow's algorithm to find a matching on rank-1 edges whereas we use Ford-Fulkerson max-flow algorithm. Gabow's algorithm runs in time $O(\sqrt{C}m)$ where $C = \sum_{i=1}^{|\mathcal{B}|} \mathsf{cap}(b_i)$ whereas since the value of max-flow in the graph $F(G_1)$ is upper bounded by $|\mathcal{A}| = n_1$, Ford-Fulkerson algorithm takes $O(mn_1)$ time. Also, the total time taken by our algorithm to partition vertices into $\mathcal{O}, \mathcal{E}$, and $\mathcal{U}$ is $O(m+n)$ where n denotes the total number of vertices in G. It is easy to see that the time spent by our algorithm in the second stage is also $O(mn_1)$ since it takes $O(m)$ time to build the tree T_a and there are at most n_1 such trees that we build. We can now conclude the following theorem.

Theorem 6. *There exists an $O(mn_1)$ time algorithm to decide whether a given instance G of the min-cost popular matchings problem admits a popular matching and if so, to compute one with min-cost.*

Note that by assigning a huge cost $\hat{C} > \sum_b \mathsf{cap}(b).\mathsf{cost}(b)$ to each of the last items $\ell_a, a \in \mathcal{A}$ that we introduced, our algorithm also works for the maximum-cardinality min-cost popular matching problem where we seek among all popular matchings of maximum cardinality, the one with minimum cost.

References

1. D. J. Abraham, K. Cechlárová, D. F. Manlove, and K. Mehlhorn. Pareto-optimality in house allocation problems. In *Proceedings of 15th Annual International Symposium on Algorithms and Computation*, pages 3–15, 2004.
2. D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.
3. P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Sciences*, 20:166–173, 1975.
4. R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Rank-maximal matchings. *ACM Transactions on Algorithms*, 2(4):602–610, 2006.
5. T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 574–584, 2009.
6. T. Kavitha and M. Nasre. Note: Optimal popular matchings. *Discrete Applied Mathematics*, 157(14):3181–3186, 2009.
7. T. Kavitha and M. Nasre. Popular matchings with variable job capacities. In *Proceedings of 20th Annual International Symposium on Algorithms and Computation*, pages 423–433, 2009.
8. M. Mahdian. Random popular matchings. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 238–242, 2006.
9. D. Manlove and C. Sng. Popular matchings in the capacitated house allocation problem. In *Proceedings of the 14th Annual European Symposium on Algorithms*, pages 492–503, 2006.
10. R. M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of the 15th Latin American Symposium on Theoretical Informatics*, pages 593–604, 2008.
11. E. McDermid and R. W. Irving. Popular matchings: Structure and algorithms. In *Proceedings of 15th Annual International Computing and Combinatorics Conference*, pages 506–515, 2009.
12. J. Mestre. Weighted popular matchings. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, pages 715–726, 2006.
13. T. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

Appendix

Proof of Property (∗) (from Section 4.3)

If $\mathcal{I}$ is a yes instance:

If $\mathcal{I}$ has a 1-in-3 satisfying assignment, then H can be augmented at a cost of exactly m to admit a popular matching.

This is similar to the proof of Lemma 5. For each j, where $1 \leq j \leq n$, do the following: if $X_j = true$, then set $\mathsf{cap}(u_j) = 1 + c_j$, where c_j is the number of clauses in which X_j is present. Else set $\mathsf{cap}(u_j) = 1$. The total cost involved here is $\sum_{j:X_j=true} c_j$. Since each clause has exactly one variable set to true, we have: $\sum_{j:X_j=true} c_j = m$. Thus the cost of our instance $\tilde{H}$ is m.

It is easy to show that the graph $\tilde{H}'$ admits an $\mathcal{A}$-perfect matching.

- Consider the people $x_1, \ldots, x_n$. Each x_j gets matched to her f-item u_j.
- Consider the people in A_i. We know that exactly one amongst $u_{j_1}, u_{j_2}, u_{j_3}$ has its capacity greater than 1 (since our assignment of capacities was based on a satisfying assignment for 1-in-3 SAT). If $\mathsf{cap}(u_{j_k}) > 1$, then a^i_k gets matched to u_{j_k} and the 2 people in $\{a^i_1, a^i_2, a^i_3\} \setminus \{a^i_k\}$ get matched to p_i and q_i. For each of the m triplets that we have here, we do as follows. The vertex a^i_{3t+k} gets matched to her top choice item r^i_t whereas the 2 people in $\{a^i_{3t+1}, a^i_{3t+2}, a^i_{3t+3}\} \setminus \{a^i_{3t+k}\}$ get matched to their last resort items.

Thus a capacity of 1 of each internal item and a capacity of exactly $\mathsf{cap}(u_j)$ of each u_j gets used. This proves that H can be augmented at a cost of exactly m to admit a popular matching.

If $\mathcal{I}$ is a no instance:

If $\mathcal{I}$ is a no *instance for 1-in-3 SAT, then H needs a cost of at least $m^3 + 1$ to admit a popular matching.*

Suppose H can be augmented a cost of at most m^3 to admit a popular matching. We will show that this translates to a 1-in-3 satisfying assignment for $\mathcal{I}$. Let $\tilde{H}$ denote the augmented graph. Let us set the truth values of variables in $\mathcal{I}$ as follows. Set $X_j = true$ iff $\mathsf{cap}(u_j)$ in $\tilde{H}$ is greater than 1.

We have only m^3 units available to augment all capacities so that people in each set A_i have items in $\tilde{H}'$ to match themselves to. Recall that the cost of each internal item is m^3. Hence it is easy to see that we cannot afford an extra copy of any internal item and thus at least one public item in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ should have its capacity more than 1 to match all of a^i_1, a^i_2, a^i_3. Otherwise there have only 2 items p_i and q_i for these 3 people to be matched to since the first copies of $u_{j_1}, u_{j_2}, u_{j_3}$ will be matched to $x_{j_1}, x_{j_2}, x_{j_3}$, respectively. Thus we have shown that at least one of $u_{j_1}, u_{j_2}, u_{j_3}$ has capacity more than 1. Hence in our assignment of truth values, there is at least 1 variable in each clause that is set to true.

Suppose 2 or more of the items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ have capacity more than 1. We have two people in $\{a^i_1, a^i_2, a^i_3\}$ having their most preferred even vertex in $\tilde{H}_1$ as an item in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$. In addition, in each of the $m^3 + 1$ triplets, two people have their most preferred even item in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$. Although, one of these 2 people from each triplet can be matched to her unique top choice item, we still need to spend $m^3 + 1$ for all the people in A_i to be matched to items in $\tilde{H}'$. This contradicts the hypothesis that H can be augmented a cost of at most m^3 into $\tilde{H}$. Hence for each i, there is exactly 1 item in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ whose capacity in $\tilde{H}$ is larger than 1. In other words, for each i, there is exactly 1 true variable in the i-th clause. Thus our assignment is a 1-in-3 satisfying assignment for $\mathcal{I}$. □

Missing details from Section 5

The first stage of our min-cost popular matching algorithm. After computing a maximum matching in G_1 (via a max-flow computation in $F(G_1)$), we partition $\mathcal{A} \cup \mathcal{B}$ into $\mathcal{O}$ (odd), $\mathcal{E}$ (even), and $\mathcal{U}$ (unreachable) as follows: initially $\mathcal{O} = \mathcal{E} = \mathcal{U} = \emptyset$.

(i) First, each person a that is unmatched in M_0 is added to $\mathcal{E}$. Also, every item b that is not fully matched in M_0 (i.e., b is matched to fewer than $\mathsf{cap}(b)$ many vertices) is added to $\mathcal{E}$ since if we had

made $\mathsf{cap}(b)$ many copies of b, then some of the copies would have been unmatched by M_0 and the other copies, which are matched, would be connected by even length alternating paths from these unmatched vertices.

(ii) Initially all vertices are unmarked. For each unmatched person/not fully matched item u, build a Hungarian tree T_u rooted at u as described below:

- For $u \in \mathcal{A}$, the children of u in T_u are all the neighboring items of u that are unmarked so far; for each of these vertices b, the children of b in T_u are all the unmarked people matched to b, the children of these people are their neighboring items that are unmarked so far and so on. As soon as a vertex is visited in T_u, we mark it so that each vertex occurs at most once in T_u.
- For $u \in \mathcal{B}$, the children of u are all the neighboring unmarked people of u (some of these could be matched to u, some of these are not – however we include all of them here since we are simulating the Hungarian tree rooted at an *unmatched* copy of u).
  Each person a in this child list has a unique child, which is the item that a is matched to. If this item is marked, then a is just a leaf in this tree, else we add $M_0(a)$ to the tree T_u, mark it and all the unmarked neighbors of $M_0(a)$ form $M_0(a)$'s children. Thus we might see some of the other people that $M_0(a)$ is matched to as its children, however since their partner in M_0 is already visited (hence marked), such people form leaf nodes in T_u.
- While building a tree T_u, we explore the neighborhood of a vertex only if this vertex is *unmarked* and then this vertex immediately gets marked. This ensures that a vertex occurs just once across all T_u's.
- Once T_u is built, all vertices that belong to even levels of T_u (the root is at level 0) are added to $\mathcal{E}$ and all vertices that belong to odd levels are added to $\mathcal{O}$.

(iii) Once we finish building all the trees T_u, where u is an unmatched person/not a fully matched item, the set $\mathcal{U}$ gets set to the vertices of $\mathcal{A} \cup \mathcal{B} \setminus \mathcal{O} \cup \mathcal{E}$ as there is no alternating path from an unmatched vertex to such vertices.