

Popularity at Minimum Cost^{*}

Telikepalli Kavitha¹, Meghana Nasre², Prajakta Nimbhorkar³

¹ Tata Institute of Fundamental Research, India. kavitha@tcs.tifr.res.in

² Indian Institute of Science, India. meghana@cse.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 $\text{cost}(b)$, that is, for any item b , new copies of b can be added to the input graph by paying an amount of $\text{cost}(b)$ 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 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, where m is the number of edges.

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 $E = E_1 \cup \dots \cup E_r$ (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,

^{*} This work was done as part of the DST-MPG partner group “Efficient Graph Algorithms”.

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 a person 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 Gärdenfors [3] in the context of stable matchings for two-sided preference lists (here both sides of the graph G express preferences). In the domain of one-sided preference lists, popular matchings can be considered to be stable as no majority vote of people can force a migration to another matching; also, popularity is based on *relative* ranking rather than the actual ranks used by people. These properties make popularity 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 $\text{cost} : \mathcal{B} \rightarrow \mathbb{R}^+$, where $\text{cost}(b)$ 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 $\text{cost}(b)$ 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. In order to minimize the cost, we will make a new copy of that item in $\{b_1, b_2\}$ which has lower cost. 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 $\text{cost}(b) \geq 0$ 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, $\sum_{b \in \mathcal{B}} \text{cost}(b)$, 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 person 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 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 $\langle c_1, \dots, c_{|\mathcal{B}|} \rangle$ denoting upper bounds on the capacities of each job. The problem is to determine if there exists $(x_1, \dots, x_{|\mathcal{B}|})$ such that for each i , setting the capacity of the 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 $\text{copies}(b)$ 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* number of copies denoted by $\text{copies}(b)$ 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 $k \cdot \text{cost}(b)$ if k copies of item b are used in M , where $k \leq \text{copies}(b)$). 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 considered this problem without costs in the context of House Allocation. There items were called houses and copies of items as in our case were represented using capacities for houses. 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. Manlove and Sng [9] showed an $O((\sqrt{C} + n_1)m)$ algorithm for the CHAT problem, where C is the sum of capacities of all items. 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, several variants of the popular matchings problem have been 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 studied 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 considered the min-cost popular matchings but in this version the costs are associated with edges whereas in our problem 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 person 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 $\text{cost}(b) \geq 0$. The problem is to “construct” a graph G or equivalently, set suitable values for $\text{copies}(b)$ where $b \in U$, 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, \dots, C_m be the clauses in \mathcal{I} and let X_1, \dots, 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 $A_i = \{a_1^i, \dots, a_9^i\}$. 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 items also appear on the preference lists of people in A_i as well as outside A_i . 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 in Fig. 1.

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 set \mathcal{A} of people in our instance is $\cup_i A_i$. The universe U of all items is the union of $\{u_1, \dots, u_n\}$ (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 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 $j = 1, \dots, n$, is 3 units.

Recall that our problem is to determine a set \mathcal{B} of items with suitable copies 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 $\text{copies}(q^i) = 3$ 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 non-zero copies 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.⁴ Thus we have the following options:

- (i) set $\text{copies}(u_{j_1}) = 0$ 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 $\text{copies}(p_1^i) = 2$
- (iii) add one more copy of u_{j_1} and set $\text{copies}(p_1^i) = 0$ and thus match a_4^i to u_{j_1} and a_7^i to q^i

⁴ A matching that contains the 3 edges $(a_1^i, u_{j_1}), (a_4^i, p_1^i), (a_7^i, q^i)$ 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.

It is not possible to have option (i) for all the u_j 's since we are forced to have non-zero copies 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. \square

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 exists a 1-in-3 satisfying assignment for \mathcal{I} .*

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, \dots, 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 $X_i = \text{true}$ then set $\text{copies}(u_i) = 0$, else $\text{copies}(u_i)$ 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 copies of every item, the corresponding u_{j_k} has 0 copies. 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 exactly 1 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 $j = 1, \dots, n$ set $X_j = \text{true}$ if and only if $\text{copies}(u_j) = 0$. 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 at least 2 items to have non-zero copies 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 zero copies. 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 non-zero copies, 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 at least one copy, 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 $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ which has zero copies. This finishes the proof of this lemma. \square

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 $\text{cost}(b)$ associated with it. The problem is to determine how to make extra copies of items in \mathcal{B} so that the resulting graph admits a popular matching and the cost of the extra copies 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 which items should be duplicated and how many extra copies should be made. Since G' does not admit a popular matching, there exists a set S of people such that the neighborhood $N(S)$ of S in G' satisfies $|N(S)| < |S|$. Let S denote a minimal such set of people. It is easy to see that every $a \in S$ must be even in G_1 . Thus, for each $a \in S$, the edge (a, s_a) belongs to G' and it must be that $s_a = z_a$. 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 be always matched in any maximum cardinality matching in G' . Hence, such an a cannot belong to S due to its minimality. Further note that for any such minimal set S , the set $N(S)$ is a set of items that are all *odd* in the graph G' with respect to a maximum cardinality matching in G' .

Since $s_a = z_a$ for every $a \in S$, and the preference lists are of length at most 2, there are no items sandwiched between $f(a)$ and $s(a)$ in a 's preference list for every $a \in S$. Thus, in order to ensure that these people get matched in any popular matching, we need to make extra copies of items in $N(S)$ or equivalently items that are *odd* in the graph G' . Our algorithm precisely does this and in order to get a min-cost augmentation, it chooses the odd item in G' which has least cost. The steps of our algorithm are described in Algorithm 1.

Our algorithm maintains the invariant that no person a changes her s -item due to the increase in copies. This is because we ensure that no top choice item b ever becomes *even* in H_i^1 , the graph H_i restricted to rank-1 edges. Note that the set of *odd* items in H_i is identified by constructing alternating paths from a person who is unmatched in H_i and every item b that appears on such a path is always *odd*. Further, our duplications ensure that the total number of copies of an item b in any augmented instance H_i is bounded by the degree of b in G' . In the case of a top choice item b , the degree of b in G' is equal to the degree of b in G_1 , the graph G restricted to rank-1 edges. Thus, even with the extra copies, a top choice item remains critical in the augmented graph restricted to rank-1 edges. This implies that for every person, the most preferred *even* item in the augmented graph restricted to rank-1 edges (i.e., its s -item) remains unchanged.

We note that the above claim also implies that in every iteration of the while loop in Step 4 of our algorithm, the size of the maximum cardinality matching

Algorithm 1 Min-cost augmentation for strict lists of length at most 2.

- 1: Construct the graph $G' = (\mathcal{A} \cup \mathcal{B}, E')$ where $E' = \{(a, b) : a \in \mathcal{A}, b \in f(a) \cup s(a)\}$.
 - 2: $H_0 = G$, $H'_0 = G'$, Let M_0 denote a maximum cardinality matching in H'_0 .
 - 3: $i = 0$.
 - 4: **while** M_i is not \mathcal{A} -complete matching **do**
 - 5: Partition the vertices into *odd* (\mathcal{O}), *even* (\mathcal{E}), *unreachable* (\mathcal{U}) w.r.t. M_i .
 - 6: Let b denote the cheapest item in $\mathcal{B} \cap \mathcal{O}$.
 - 7: Set $\text{copies}(b) = \text{copies}(b) + 1$.
 - 8: Construct the graph H'_{i+1} corresponding to H_{i+1} and update M_{i+1} to be a maximum cardinality matching in H'_{i+1} .
 - 9: $i = i + 1$.
 - 10: **end while**
 - 11: Output the graph H_i .
-

increases by 1, that is, $|M_{i+1}| = |M_i| + 1$. Therefore, the while loop terminates in $k = |\mathcal{A}| - |M_0|$ iterations. Since k is bounded by n_1 , the number of applicants in G , the running time of our algorithm is $O(n_1^2)$. It is clear that the graph H_i returned by the algorithm admits an \mathcal{A} -complete matching the graph H'_i and hence admits a popular matching. To see that the instance returned is a min-cost instance, observe that there is no alternating path between an item b which got duplicated in our algorithm and an item b' whose cost is strictly smaller than the $\text{cost}(b)$. Otherwise in the iteration when b was *odd*, so was b' and it would have been picked up by our algorithm.

We can therefore conclude 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, \dots, C_m be the clauses in \mathcal{I} and let X_1, \dots, 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.

The set \mathcal{B} of items is the union of $\cup_{i=1}^m D_i$ (the set of all the internal items) and $\{u_1, \dots, u_n\}$ (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 $\cup_{i=1}^m A_i$

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

and $\{x_1, \dots, x_n\}$, 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 every item has a single copy, 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 such that \tilde{G} admits a popular matching. We now state the following lemma that establishes the reduction.

Lemma 4. \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 copies of the items in the following manner: the copies of the internal items remain the same, i.e., $\text{copies}(b) = 1$ for each $b \in \cup_i D_i$ and the copies of the public items are set as follows.

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

- if $X_j = \text{true}$, then set $\text{copies}(u_j) = 1 + c_j$
- else $\text{copies}(u_j)$ 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: $\sum_{j: X_j = \text{true}} c_j = m$. Thus the cost of our augmentation is m .

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

- Consider the people x_1, \dots, 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 more than one copies (since the number of copies was based on a satisfying assignment for 1-in-3 SAT). If $\text{copies}(u_{j_k}) > 1$, 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 $\text{copies}(u_j) > 1$ in \tilde{G} , then set variable $X_j = \text{true}$, else set $X_j = \text{false}$. We need to show that this is a 1-in-3 satisfying assignment for \mathcal{I} .

Since the cost of adding one copy 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 copies of items have been added 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 $\text{copies}(u) > 1$ 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 $\text{copies}(u) > 1$ 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 more than 1 copies. 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 add an extra copy of some 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} . \square

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 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 and 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_i^t	u_{j_1}
a_{3t+2}^i	r_i^t	u_{j_2}
a_{3t+3}^i	r_i^t	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 $\{x_1, \dots, x_n\}$. Since each clause has 3 variables, $n \leq 3m$. Thus we can bound n_1 , the number of people in H as: $n_1 \leq 3m^4 + 9m \leq 4m^4$ for $m \geq 3$.

Recall that for each j , 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_i^k , for $k = 1, \dots, m^3 + 1$ is m^3 , and the cost of each u_j for $j = 1, \dots, n$ is 1. We now show that the instance constructed as above satisfies the property (*).

Lemma 5. *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.*

Proof. We first consider the case when \mathcal{I} is an *yes* instance. The proof is similar to that of Lemma 4. For each j , where $1 \leq j \leq n$, do the following: if $X_j = \text{true}$, then set $\text{copies}(u_j) = 1 + c_j$, where c_j is the number of clauses in which X_j is present. Else set $\text{copies}(u_j) = 1$. The total cost involved here is $\sum_{j: X_j = \text{true}} c_j$. Since each clause has exactly one variable set to true, we have: $\sum_{j: X_j = \text{true}} c_j = m$. Thus, the cost of our instance \tilde{H} is m . It is easy to show that the graph \tilde{H}' admits an \mathcal{A} -complete matching.

- Consider the people x_1, \dots, 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 more than one copies (since the number of copies was based on a satisfying assignment for 1-in-3 SAT). If $\text{copies}(u_{j_k}) > 1$, 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 . For each of the $m^3 + 1$ triplets that we have here, we do as follows. The person a_{3t+k}^i gets matched to her top choice item r_i^t whereas the 2 people in $\{a_{3t+1}^i, a_{3t+2}^i, a_{3t+3}^i\} \setminus \{a_{3t+k}^i\}$ get matched to their last items.

This proves that H can be augmented at a cost of exactly m to admit a popular matching.

We now prove the other direction, that is, 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 at 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 $X_j = \text{true}$ iff $\text{copies}(u_j)$ in \tilde{H} is greater than 1.

We have only m^3 units available to make extra copies 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 more than one copy to match all of a_1^i, a_2^i, a_3^i . Otherwise there are 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 more than one copy. 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 more than one copy in \tilde{H} . We have two people in $\{a_1^i, a_2^i, a_3^i\}$ having their most preferred *even* item 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 $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ that has more than one copy in \tilde{H} . 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} . \square

Now suppose that 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 $1/2 \cdot \sqrt{4m^4} \cdot m = m^3$. 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 using **Algo1** it is possible to determine whether \mathcal{I} has a 1-in-3 satisfying assignment or not, a contradiction. Hence we conclude the following theorem.

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

4.4 Perfect augmentation

In this section we consider a variant of the min-cost augmentation problem where we are not content with only a popular matching, but insist on every applicant being matched to an item. Note that till now we allowed people to be matched to their last items which was equivalent to leaving them unmatched. We call a popular matching that matches all the people as a *perfect popular matching* and denote the problem as *min-cost perfect augmentation*. We show that this problem becomes NP-hard even when preference lists are strict and have length at most 2 in contrast to the min-cost augmentation problem which has a simple polynomial time algorithm for strict lists of length at most 2.

The overall reduction is similar to the min-cost augmentation problem where we reduce from an instance \mathcal{I} of monotone 1-in-3 SAT problem. As earlier, corresponding to every variable X_j in \mathcal{I} we have an applicant x_j in our instance. The preference list of applicant x_j contains only one item u_j . Let $C_i = (X_{j_1} \vee X_{j_2} \vee X_{j_3})$ be a clause in \mathcal{I} . Corresponding to clause C_i we have a set of 6 people $A_i = \{a_1^i, \dots, a_6^i\}$ and a set of internal items $D_i = \{p_i, q_i\}$. The internal items appear on preference lists of people only in A_i . In addition public items $u_{j_1}, u_{j_2}, u_{j_3}$ also appear on the preference lists of people in A_i . The preference lists of people are as shown in Fig. 4.

a_1^i	p_i	u_{j_1}		a_4^i	u_{j_1}	q_i	
a_2^i	p_i	u_{j_2}		a_5^i	u_{j_2}	q_i	
a_3^i	p_i	u_{j_3}		a_6^i	u_{j_3}	q_i	

Fig. 4. Preference lists of the 6 people in A_i

We define the sets \mathcal{A} and \mathcal{B} below.

$$\begin{aligned} \mathcal{A} &= \cup_{i=1}^m A_i \cup \{x_1, \dots, x_n\}. \\ \mathcal{B} &= \cup_{i=1}^m D_i \cup \{u_1, \dots, u_n\}. \end{aligned}$$

The graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ with the preference lists as defined above is our instance G . It remains to describe the costs of items in G . The cost of every public item u_j is 1 whereas the costs of every internal item is m . With a single copy of every item, all public items are critical in the graph G_1 . Hence people a_1^i, a_2^i, a_3^i belonging to A_i treat their unique last resort items as their s -items. Further people a_4^i, a_5^i, a_6^i treat the internal item q_i as their s -item. It is easy to verify that the graph G described above does not admit a popular matching.

Let \tilde{G} be an augmented instance that admits a perfect popular matching. In any perfect popular matching in \tilde{G} every applicant x_j has to be matched to her f -item i.e. u_j . Now let us consider the people in A_i . Since the internal items are of cost m each and public items have cost 1, it is cheaper to make extra copies of public items. Further, recall that any popular matching has to be a

maximum cardinality matching on rank-1 edges. Therefore in order to match one of $\{a_1^i, a_2^i, a_3^i\}$ (say a_1^i) to its s -item u_{j_1} , we need to make 2 extra copies of u_{j_1} thus making u_{j_1} non-critical in \tilde{G}_1 . This ensures that applicant a_4^i gets matched to a copy of u_{j_1} , her f -item. It is easy to see that in order to get a perfect popular matching, we need to make at least two public items non-critical in the graph \tilde{G} . Thus we spend an amount of at least 4 units per clause in any augmented instance \tilde{G} that admits a perfect popular matching.

We now show the following lemma which proves the correctness of our reduction.

Lemma 6. *The min-cost instance \tilde{G} corresponding to G that admits a perfect popular matching has cost $4m$ iff there exists a 1-in-3 satisfying assignment for \mathcal{I} .*

Proof. Assume that there exists a 1-in-3 satisfying assignment for \mathcal{I} and let $Truthval$ denote the 1-in-3 satisfying assignment for \mathcal{I} . For every variable X_j , let c_j denote the number of clauses in which X_j appears. We set the copies of the items in \tilde{G} as follows: If $Truthval(X_j) = false$, then $\text{copies}(u_j) = 2c_j + 1$, otherwise $\text{copies}(u_j) = 1$. Thus we pay $2c_j$ units for the extra copies of item u_j whenever the corresponding variable X_j is set to false.

Since \mathcal{I} has a 1-in-3 satisfying assignment, each clause has exactly two variables set to false. We therefore have:

$$\sum_{j: Truthval(X_j)=false} 2c_j = 4m.$$

Thus the cost of our instance \tilde{G} is $4m$. We now show that in the graph \tilde{G} , every applicant has an item to match amongst her f or s items.

- Consider the people x_1, x_2, \dots, x_n . Each x_j gets matched to her f -item u_j .
- Consider the people in A_i corresponding to the clause $C_i = (X_{j_1} \vee X_{j_2} \vee X_{j_3})$. We know that exactly two among $u_{j_1}, u_{j_2}, u_{j_3}$ are non-critical. Assume that u_{j_1} and u_{j_2} are non-critical, then people a_1^i and a_2^i get matched to their s -items u_{j_1} and u_{j_2} respectively. Further a_4^i and a_5^i get matched to their f -items u_{j_1} and u_{j_2} respectively. Finally, a_3^i gets matched to her f -item p_i whereas a_6^i gets matched to her s -item q_i .

To prove the other direction, assume that \tilde{G} admits a popular matching and has cost $4m$. We now translate this into the truth values for variables in \mathcal{I} . If $\text{copies}(u_j) > 1$ in \tilde{G} , then set variable $X_j = false$ else set $X_j = true$. We need to show that this assignment is a 1-in-3 satisfying assignment.

Recall that for every clause we need to spend at least 4 units in order to match all the people in A_i . Since the cost of \tilde{G} is $4m$, this implies that exactly 4 units have been spent per clause for making extra copies.

Now consider the people in A_i . The internal items occurring on their preference lists of these people have cost m each, hence none of the internal items have extra copies in the augmented instance. This implies that at least 2 public items

amongst $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ have more than one copy in the augmented instance. Thus our truth assignment sets at least 2 variables to *false*. It remains to prove that for every clause exactly 2 variables are set to *false*. Assume not.

Let C_i be a clause in which our truth assignment set all the three variables $X_{j_1}, X_{j_2}, X_{j_3}$ to *false*. This implies that in the instance \tilde{G} all the three public items $u_{j_1}, u_{j_2}, u_{j_3}$ have more than one copy. The first copy of these public items is utilized in matching people $x_{j_1}, x_{j_2}, x_{j_3}$ respectively to their f -items. Further since any popular matching is a maximum cardinality matching on rank-1 edges, the second copies of these items will have to be assigned to people a_4^i, a_5^i, a_6^i . Thus after spending 3 units we are still left with 2 people amongst $\{a_1^i, a_2^i, a_3^i\}$ without any item. Hence we need to spend 2 more units to match 2 of these people, which implies that we spent $3+2=5$ units for people in A_i . This contradicts the fact that exactly 4 units was spent in order to match people in A_i , for all i . Thus exactly one variable in every clause is set to *true*. \square

The following theorem is immediate from the above lemma.

Theorem 6. *The min-cost perfect augmentation problem is NP-hard for strict lists of length at most 2.*

5 Min-cost Popular matchings

In this section we present an $O(mn_1)$ time algorithm for the min-cost popular matchings problem, where $m = |E|$ and $n_1 = |\mathcal{A}|$. Our input is an instance $G = (\mathcal{A} \cup \mathcal{B}, E)$ where each item $b \in \mathcal{B}$ has associated with it the number $\text{copies}(b)$ (denoting the maximum number of people that can be matched to b) and a price $\text{cost}(b) \geq 0$. Whenever a person gets matched to b , an amount of $\text{cost}(b)$ has to be paid. Thus if $k \leq \text{copies}(b)$ copies of b gets used in a matching M , then a cost of $k \cdot \text{cost}(b)$ 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 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 $\text{copies}(b)$ copies of each vertex b , call them $b_1, \dots, b_{\text{copies}(b)}$, 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 $\text{copies}(b)$ 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 $\text{copies}(b)$ many duplicates of every item b in G our problem becomes equivalent to the original popular matchings problem.

5.1 Our algorithm

Our algorithm to compute a min-cost popular matching can be broadly partitioned into two stages. In the first stage we build the graph G' , i.e. the graph where every person adds edges to their f and s -items. Identifying s -items for people involves partitioning the vertices of G into *odd*, *even* and *unreachable* with respect to a maximum cardinality matching on rank-1 edges. We show how to efficiently do this by building Hungarian trees rooted at unmatched vertices. The second stage then computes a min-cost popular matching in the graph G' if one exists.

The first stage. 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 $\text{copies}(b)$ 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 person is matched to at most 1 item and every item b is matched up to $\text{copies}(b)$ people. A maximum flow in $F(G_1)$ becomes a maximum cardinality matching in G_1 . We 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 , our goal is to obtain a partition of $\mathcal{A} \cup \mathcal{B}$ into \mathcal{O} (*odd*), \mathcal{E} (*even*) and \mathcal{U} (*unreachable*). This can be done in $O(\text{number of edges})$ provided we create $\text{copies}(b)$ many duplicates of each item p and duplicate the neighborhood of b for each copy of b . However this is too expensive. The main point to note is that all the $\text{copies}(b)$ many copies of b , for each item b , have the same *odd/even/unreachable* status. We show below that we can remain in the graph G_1 and determine the *odd/even/unreachable* status of all the vertices in linear time.

1. We begin with $\mathcal{O} = \mathcal{E} = \mathcal{U} = \emptyset$.
2. We then add to the set \mathcal{E} all people that are unmatched in M_0 and all items that are not fully matched by M_0 (i.e. a item b that is matched to fewer than $\text{copies}(b)$ many people). This is because if we would have made $\text{copies}(b)$ many duplicates of b , some of the copies would have remained unmatched by M_0 and the other copies which are matched would be connected by *even* length alternating paths from these unmatched vertices.

3. Our goal now is to build a Hungarian tree T_u for each vertex u that is unmatched or not fully in M_0 . In order to do so we first set all vertices as unmarked. We build the trees rooted at unmatched people and not fully matched items as described below:
 - (a) For $u \in \mathcal{A}$ that is unmatched, the children of u in T_u are all the neighboring items of u that are unmarked so far. For each of these items b the children of b in T_u are all the unmarked people matched to b . The children of these people are their neighboring unmarked items and so on. As soon as a vertex gets visited in T_u we mark it.
 - (b) For $u \in \mathcal{B}$ the children of u are all the neighboring unmarked people of u . Note that some of these people could be matched to u – however, we will include all these people since we are simulating the Hungarian tree rooted at an *unmatched* copy of u . We mark each person in this child list.
Each person a in the above child list had a unique child, the item to which a is matched. If this item is marked, then a is a leaf in this tree, else we add $M_0(a)$ to the tree and mark it. We now continue to explore the unmarked neighborhood of $M_0(a)$ for all non-leaf people a .
 - (c) 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} .
4. Once we finish building all the trees T_u , where u is 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.

We note that 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. Having obtained the partition, it is now possible to define $s(a)$ for every person a as the most preferred *even* item of a . Let the graph G' be the graph G_1 along with the edges (a, b) where $a \in \mathcal{E}$ and $p \in s(a)$.

Since a popular matching is a maximum cardinality matching on rank-1 edges, all items that are *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 with how many copies in the min-cost popular matching. We make this choice in the second stage of our algorithm.

The second stage. Our goal in the second part of the algorithm is to augment the matching M_0 to find a min-cost popular matching. However, we start with the matching M_1 , where $M_1 = M_0 \setminus \{\text{all edges } (a, b) \text{ where } a \in \mathcal{O}\}$. Thus M_1 consists only of edges (a, b) where $b \in \mathcal{O} \cup \mathcal{U}$. We take M_1 to be our starting matching rather than M_0 because it may be possible to match people $\mathcal{O} \cap \mathcal{A}$ to cheaper rank-1 neighbors. Recall that while computing the max-flow M_0 , the costs of items played no role.

Now let ρ be an augmenting path with respect to M_1 , i.e., one end of ρ is an unmatched person and the other end of ρ is item b that is not fully matched. The cost of augmenting the current matching along ρ is the cost of b . By augmenting the current matching along ρ , every item other than b that is currently matched stays matched to the same number of people and the item b gets matched to one more person. Thus the cost of the new matching is the cost of the old matching + $\text{cost}(b)$. In order to match an unmatched person a , our algorithm always chooses the cheapest augmenting path starting from the person a .

To find the cheapest augmenting path we build a Hungarian tree T_a rooted at every person that is unmatched in M_1 . Initially all vertices are unmarked and while building T_a every visited vertex gets marked so that each vertex occurs at most once in T_a . We do not terminate the construction of T_a as soon as we find an augmenting path, but we build T_a completely in order to find a min-cost item b such that there is an augmenting path between a and b ; we augment M_1 along this path to obtain M_2 . On the other hand if T_a has no augmenting path then we quit and declare “ G does not admit a popular matching”.

Algorithm 2 Algorithm for min-cost popular matching.

- 1: Construct the graph $G_1 = (\mathcal{A} \cup \mathcal{B}, E_1)$ where $E_1 = \{(a, b) : a \in \mathcal{A}, b \in f(a)\}$.
 - 2: Construct the flow graph $F(G_1)$ by adding two vertices s and t and adding directed edges with appropriate capacities.
 - 3: Compute a maximum flow in $F(G_1)$ and translate the flow to a matching M_0 in G_1 .
 - 4: Obtain a partition of the vertices of G as *odd* (\mathcal{O}), *even* (\mathcal{E}) and *unreachable* (\mathcal{U}) using M_0 .
 - 5: Construct the graph $G' = (\mathcal{A} \cup \mathcal{B}, E')$ where every person adds edges to her f -items and every *even* person adds edges to her s -items.
 - 6: Delete from G' all $\mathcal{O}\mathcal{O}$ and $\mathcal{O}\mathcal{U}$ edges.
 - 7: Delete from M_0 all edges that are incident on *odd* people in G' and call the resulting matching M_1 .
 - 8: $i = 1$.
 - 9: **while** there exists an unmatched person a in M_i **do**
 - 10: Build a Hungarian tree T_a rooted at a .
 - 11: **if** there exists no augmenting path starting at a **then**
 - 12: Quit and declare “ G does not admit any popular matching”.
 - 13: **else**
 - 14: Augment M_i along the cheapest augmenting path in T_a and call the new matching M_{i+1} .
 - 15: **end if**
 - 16: $i = i + 1$.
 - 17: **end while**
 - 18: Return M_i .
-

We present our entire algorithm in Algorithm 2. To see the correctness of the algorithm we first note that if there is no augmenting path in T_a , where a is an unmatched person in M_i , 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 a item in $f(a) \cup s(a)$. It remains to prove that if G admits a popular matching, then the matching $M(= M_i)$ returned at Step 18 of Algorithm 2 is a min-cost popular matching. We prove that using Lemma 7.

Lemma 7. *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 $\text{copies}(b)$ many copies and M and OPT both refer to matchings where each item is matched to at most one person. Consider $\text{OPT} \oplus M$ - 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 $\text{OPT} \oplus M$. 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 $\text{cost}(\beta_M) \leq \text{cost}(\beta_0)$. 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, $\text{cost}(\beta_1) \leq \text{cost}(\beta_0)$.

We now repeat the same argument for the β_1 - β_M sub-path of ρ . Let α_2 be the last person in the β_1 - β_M sub-path 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 $\text{cost}(\beta_2) \leq \text{cost}(\beta_1)$.

Repeating the same argument for the β_2 - β_M sub-path we get vertices $\beta_3, \dots, \beta_t = \beta_M$ where $\text{cost}(\beta_2) \leq \text{cost}(\beta_3) \leq \dots \leq \text{cost}(\beta_t)$. Combining all the inequalities yields $\text{cost}(\beta_M) \leq \text{cost}(\beta_0)$. \square

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 $C = \sum_{i=1}^{|\mathcal{B}|} \text{copies}(b_i)$ 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 7. *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 $\hat{C} > \sum_b \text{copies}(b) \cdot \text{cost}(b)$ to each of the last items $\ell_a, a \in \mathcal{A}$ that we introduced, our algorithm also works for the min-cost maximum-cardinality popular matching problem where we seek among all popular matchings of maximum cardinality, the one with minimum cost.

Conclusions

In this paper we considered several extensions of the popular matching problem. We showed that the min-cost popular instance problem, which involves building a min-cost graph that admits a popular matching that matches all applicants, is NP-hard, even when preference lists are strict and of length at most 2. In contrast, the min-cost *augmentation* problem admits a simple polynomial time algorithm when preference lists are strict and of length at most 2. However, the min-cost augmentation problem is NP-hard in general; it is NP-hard even when preference lists are strict and of length at most 3. In fact, it is NP-hard to approximate the min-cost augmentation problem to within a factor of $\sqrt{n_1}/2$, where n_1 is the number of people. We also showed that the min-cost popular matching problem (the number of copies of each item is fixed here) can be solved in $O(mn_1)$ time, where m is the number of edges in the input graph.

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.