J Comb Optim (2014) 27:574–596
DOI 10.1007/s10878-012-9537-0

Popularity at minimum cost

Telikepalli Kavitha · Meghana Nasre · Prajakta Nimbhorkar

Published online: 21 August 2012

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 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

A preliminary version of this work appeared in 21st International Symposium on Algorithms and Computation, ISAAC'10 (Kavitha et al. 2010).

This work was done when the second author was a student at the Indian Institute of Science, Bangalore, India and the third author was a student at the Institute of Mathematical Sciences, Chennai, India.

T. Kavitha
Tata Institute of Fundamental Research, Mumbai, India
e-mail: kavitha@tcs.tifr.res.in

M. Nasre (✉)
Department of Computer Science, The University of Texas at Austin, 1616 Guadalupe, Suite 2.408, Austin, TX 78701, USA
e-mail: meghana@cs.utexas.edu

P. Nimbhorkar
Chennai Mathematical Institute, Siruseri, India
e-mail: prajakta@cmi.ac.in

$\sqrt{n_1}/2$, 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.

Keywords Bipartite graphs · Matchings · One-sided preference lists · NP-hardness

1 Introduction

The *popular matching* problem deals with matching people to items, where each person ranks a subset of items in 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 (Abraham et al. 2004), rank-maximality (Irving et al. 2006), 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 greater than those that prefer M' to M. A matching M^* is *popular* if there is no matching that is more popular than M^*.

The notion of popularity is an appealing notion of optimality since it does not use absolute ranks and further no majority vote of people can force migration to another matching. 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 the preference lists of the people over the items are as shown in Fig. 1. That is, each person 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. (2007) designed efficient algorithms for determining if a given instance admits a popular matching and computing one, if it exists.

a_1	b_1	b_2	b_3
a_2	b_1	b_2	b_3
a_3	b_1	b_2	b_3

Fig. 1 Example instance that does not admit a popular matching

The fact that popular matchings do not always exist has motivated several extensions to the popular matching problem, see McCutchen (2008), Kavitha et al. (2011), Kavitha and Nasre (2011). In this paper we study two further generalizations namely *min-cost augmentation problem* and *min-cost popular instance*. In the min-cost augmentation problem our goal is to *augment* the input graph such that then new graph admits a popular matching. In the min-cost popular instance problem our goal is to *construct* an instance that admits a popular matching.

1.1 Min-cost augmentation

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 back to the earlier example on 3 people and 3 items (as shown in Fig. 1) 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.

1.2 Min-cost popular instance

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 such that the number of people who are happier after the reassignment exceeds the number who are unhappier.

1.3 Our results

We show the following results in this paper:

- 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).
- The min-cost augmentation problem has a polynomial time algorithm when each preference list has length at most 2.
- 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.

The NP-hardness results for the min-cost augmentation/min-cost popular instance problems stem from the fact that 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 $\mathsf{copies}(b)$ 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 $\mathsf{copies}(b)$ 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 $k \cdot \mathsf{cost}(b)$ if k copies of item b are used in M, where $k \leq \mathsf{copies}(b)$). 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 (2006) showed an $O(m(n_1 + \sqrt{C}))$ algorithm for the CHAT problem, where C is the sum of capacities of all items.

1.4 Background

Popular matchings were first introduced by Gärdenfors (1975) in the context of stable matchings for two-sided preference lists (here both sides of the graph G express preferences). Abraham et al. (2007) studied it in the context of one-sided preferences where only one side of the bipartition ranks the members of the other side. They gave a structural characterization of graphs that admit popular matchings and also gave efficient algorithms to compute a popular matching if one exists. Section 2 outlines this characterization and the algorithm that follows from it.

Subsequent to the work in Abraham et al. (2007), 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 (2006), the weighted version studied by Mestre (2008) and random popular matchings studied by Mahdian (2006). Kavitha and Nasre (2009) as well as McDermid and Irving (2011) 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.

The other line of research includes extensions for instances when no popular matching exists. McCutchen (2008) 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. (2011) generalized the notion of popularity to *mixed matchings* or probability distributions over matchings and showed that a popular mixed matching always exists. Kavitha and Nasre (2011) considered the problem of popular matchings with variable item copies which is closely related to the problems considered in this paper. 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 items, along with a list $\langle c_1, \ldots, c_{|\mathcal{B}|} \rangle$ denoting upper bounds on the number of copies of each item. The problem is to determine if there exists $(x_1, \ldots, x_{|\mathcal{B}|})$ such that for each i, having x_i copies of the i-th item, 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 Kavitha and Nasre (2011). We would like to contrast the NP-hardness of the min-cost augmentation problem with the problem of determining a popular matching with variable item copies. Note that in case of the popular matchings with variable copies, the number of copies of *each* item has an upper bound. Instead, if we only had to maintain an overall upper bound on the total number of copies of all the items rather than individual upper bounds, a simple polynomial time algorithm solves this problem (Kavitha and Nasre 2011).

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 Kavitha and Nasre 2011). However, when the costs come from $\{1, 2\}$ the problem becomes NP-hard.

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 Sect. 5 has our algorithm for the min-cost popular matching problem.

2 Preliminaries

We review the characterization of popular matchings given in Abraham et al. (2007). Let $G_1 = (\mathcal{A} \cup \mathcal{B}, E_1)$ be the graph containing only rank-1 edges. Then Abraham et al. (2007, Lemma 3.1) show 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, in G_1. The following lemma, proved in Pulleyblank (1995), is well known in matching theory.

Lemma 1 *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 $M \cap E_1$ in G_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 Abraham et al. 2007) *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 $\mathsf{cost}(b) \geq 0$. The problem is to "construct" a graph G or equivalently, set suitable values for $\mathsf{copies}(b)$ 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.

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

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 be true. This problem is NP-hard (Schaefer 1978).

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. 2. 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 induced by clause C_i appear only on the preference lists of a subset of the people of A_i while the public items appear on the preference lists of people in A_i as well as outside A_i. The public item u_{j_k} corresponds to the variable X_{j_k}. In every clause C_i that X_{j_k} belongs to, the item u_{j_k} appears in the preference lists of some of the people in the set A_i as shown in Fig. 2.

The set $\mathcal{A}$ of people in our instance is $\bigcup_i A_i$. The universe U of all items is the union of $\{u_1, \ldots, u_n\}$ (the n public items) and the set $\bigcup_i \{p_1^i, p_2^i, p_3^i, q^i\}$ 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 $j = 1, \ldots, n$, 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 $14m$, where m is the number of clauses in the corresponding 1-in-3 SAT instance.*

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. 2. 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 people 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 $\mathsf{copies}(q^i) = 3$ 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. Consider the people in the set $S_k = \{a_k^i, a_{k+3}^i, a_{k+6}^i\}$, for $k \in \{1, 2, 3\}$. We observe that in case u_{j_k} does not have any copy then the people in S_k can be matched by spending 4 units. That is, spend 3 units to match a_k^i and 1 unit to match a_{k+3}^i to a copy of p_k^i and 0 units to match a_{k+6}^i to a copy of q^i. Note that here, a_k^i gets matched to a copy of u_{j_l} where $j \neq l$ and u_{j_l} is on the preference list of a_k^i. However, when u_{j_k} has non-zero copies we claim that we have to spend at least 5 units in order to match the people in S_k in any popular matching. With non-zero copies of u_{j_k}, we have the following options to match the people in S_k:

(i) Match a_k^i and a_{k+3}^i to two copies of u_{j_k} and match a_{k+6}^i to a copy of q^i. This costs us 6 units.
(ii) Match a_k^i to a copy of u_{j_k} and match a_{k+3}^i and a_{k+6}^i to two copies of p_k^i. This costs us 5 units.
(iii) Match a_k^i to a copy of u_{j_k}, a_{k+3}^i to a copy of p_k^i and match a_{k+6}^i to a copy of q^i. This option is the cheapest which costs us 4 units, however it is not a feasible option due the following. Recall that there are non-zero copies of the item u_{j_k} and hence p_k^i is the second choice item for a_{k+3}^i. Since p_k^i is a_{k+6}^i's top choice item, we also have to match a_{k+6}^i to p_k^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_{k+6}^i to q^i in a popular matching while p_k^i gets matched to a_{k+3}^i who regards this item as a second choice item.

It is clear from the above that, option (ii) is the cheapest amongst the feasible options. Thus when we have non-zero copies of u_{j_k} we have to spend at least 5 units in order to match all the people in S_k in a popular matching. We further note that the preference lists of the people a_1^i, a_2^i, a_3^i force us to have non-zero copies for at least 2 of the 3 items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$. This implies that in order to match all the people in A_i in any popular matching we have to spend at least $4 + 5 + 5 = 14$ units.

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 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 $X_1, \ldots, X_n$ such that each clause has exactly 1 literal set to be true (and thus 2 literals set to be 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 $X_j = \mathit{true}$ then set $\mathsf{copies}(u_j) = 0$, else $\mathsf{copies}(u_j)$ will be set to a suitable strictly positive value.

Since the setting of true/false values to X_j'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 $j = 1, \ldots, n$ set $X_j = \mathit{true}$ if and only if $\mathsf{copies}(u_j) = 0$. 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. □

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 remark

that in this case, the master list ordering is the same as sorting the items in decreasing order of their costs. 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 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 $\mathcal{A}$-complete 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 of items that are *odd* in the graph G'. Our algorithm precisely does this and in order to get a min-cost augmentation, it iteratively chooses the *odd* item in G' which has least cost. The steps of our algorithm are described in Algorithm 1.

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: $H_0 = G$, $H'_0 = G'$.
3: Let M_0 denote a maximum cardinality matching in H'_0.
4: **for** every $b \in \mathcal{B}$ **do**
5: $\quad \mathsf{copies}(b) = 1$.
6: **end for**
7: $i = 0$.
8: **while** M_i is not an $\mathcal{A}$-complete matching **do**
9: $\quad$ Partition the set of vertices into $\mathcal{O}$ (*the set of odd vertices*), $\mathcal{E}$ (*the set of even vertices*), $\mathcal{U}$ (*the set of unreachable vertices*) w.r.t. M_i in H'_i.
10: $\quad$ Let b denote the cheapest item in $\mathcal{B} \cap \mathcal{O}$.
11: $\quad$ Set $\mathsf{copies}(b) = \mathsf{copies}(b) + 1$. This defines the new graph H_{i+1}.
12: $\quad$ 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}.
13: $\quad i = i + 1$.
14: **end while**
15: Output the graph H_i.

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 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 people 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 in the graph H'_i and hence admits a popular matching. It remains to show that the instance returned by our algorithm is indeed a minimum cost instance; we prove that using Lemma 4.

Lemma 4 *The graph H returned by Algorithm 1 is a minimum cost augmentation of G that admits a popular matching.*

Proof For the sake of contradiction suppose there is an augmentation of G with smaller cost. Among all such augmentations of minimum cost, let H_{OPT} be that min-cost augmentation such that the following sum:

$$\sum_{b \in \mathcal{B}} |\text{number of copies of } b \text{ in } H - \text{number of copies of } b \text{ in } H_{OPT}|$$

is the smallest.

Since $H \neq H_{OPT}$, the number of extra copies of duplicated items in H and H_{OPT} do not match. However, note that any item b having extra copies in H_{OPT} is an item which was *odd* in G'. Because, if b were *even/unreachable* in G', we could delete the extra copies of b from H_{OPT} and get a smaller cost instance that continues to admit a popular matching. Further, it is clear that our algorithm always makes extra copies of items that were *odd* in G'.

Now, let i be the first iteration where our algorithm chooses to make a copy of an item β such that the number of copies of β in H is more than the number of copies of β in H_{OPT}. Since H_{OPT} admits a popular matching, the item β has to be *unreachable* in H'_{OPT} (since H'_{OPT} admits an $\mathcal{A}$-complete matching). It is possible for β to be *unreachable* in H'_{OPT} only if there exists some other item β' with a larger number of copies in H_{OPT} than in H and β' satisfies the following property: β' has an alternating path from β with respect to the matching M_i in the graph H'_i constructed by our algorithm.

Thus in the iteration i when β was *odd* in H'_i, so was β'. Since our algorithm chose β to duplicate, it follows that $\mathsf{cost}(\beta) \leq \mathsf{cost}(\beta')$. Thus, we could replace a copy of β' in H_{OPT} by a copy of β, thereby getting another instance K that admits a popular matching.

In case $\mathsf{cost}(\beta) < cost(\beta')$, the cost of K is less than the cost of H_{OPT}, contradicting the fact that H_{OPT} was the minimum cost instance. Thus it has to be the case that $cost(\beta) = cost(\beta')$, and so K is another minimum cost augmentation of G that admits a popular matching. Since K has one more copy of β and one less copy of β' than H_{OPT}, this contradicts the definition of H_{OPT} as that min-cost augmentation where $\sum_{b \in \mathcal{B}} |$the number of copies of b in $H -$ the number of copies of b in $H_{OPT}|$ is the smallest. This completes the proof that the graph returned by our algorithm is indeed a minimum cost augmentation of G that admits a popular matching. □

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 Sect. 3). Let

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. 3 Preference lists of the 6 people in A_i

$\mathcal{I}$ 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_k} occurs in a clause C_i, the item u_{j_k} 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. 3.

The set $\mathcal{B}$ of items is the union of $\bigcup_{i=1}^m D_i$ (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 $\bigcup_{i=1}^m A_i$ 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 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 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 number of copies of the items in the following manner: the number of copies of the internal items remain the same, i.e., $\mathsf{copies}(b) = 1$ for each $b \in \cup_i D_i$ and the number of copies of the public items are set as follows.

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

- if $X_j = \mathit{true}$, then set $\mathsf{copies}(u_j) = 1 + c_j$
- else $\mathsf{copies}(u_j)$ 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: $\sum_{j:X_j=true} c_j = m$. 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 $x_1, \ldots, x_n$. 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 more than one copies (since the number of copies was based on a satisfying assignment for 1-in-3 SAT). If $\mathsf{copies}(u_{j_k}) > 1$, then a^i_k gets matched to u_{j_k} and one of the 2 people in $\{a^i_1, a^i_2, a^i_3\} \setminus \{a^i_k\}$ gets matched to p_i while the other person gets matched to q_i. Finally, a^i_{k+3} gets matched to her top choice item r_i whereas the 2 people in $\{a^i_4, a^i_5, a^i_6\} \setminus \{a^i_{k+3}\}$ get matched to their respective 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 $\mathsf{copies}(u_j) > 1$ 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 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, by assumption we know 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. 3). 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 $\mathsf{copies}(u) > 1$ 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 $\mathsf{copies}(u) > 1$ 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 copy. Then, we have k non-critical items in $\{u_{j_1}, u_{j_2}, u_{j_3}\}$ and so we have k people in $\{a^i_4, a^i_5, a^i_6\}$ 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 $\{a^i_4, a^i_5, a^i_6\}$ 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 $\{a^i_1, a^i_2, a^i_3\}$. 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.*

a^i_{3t+1}	r^t_i	u_{j_1}	
a^i_{3t+2}	r^t_i	u_{j_2}	
a^i_{3t+3}	r^t_i	u_{j_3}	

Fig. 4 Preference lists of people corresponding to the t-th triplet

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^i_4, a^i_5, a^i_6) in Fig. 3 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 $(a^i_{3t+1}, a^i_{3t+2}, a^i_{3t+3})$ is shown in Fig. 4.

We now have $3 + 3(m^3 + 1)$ people in A_i, namely a^i_1, a^i_2, a^i_3 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 $\bigcup_i A_i$), 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 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^k_i, for $k = 1, \ldots, m^3 + 1$ is m^3, and the cost of each u_j for $j = 1, \ldots, n$ is 1. We now show that the instance constructed as above satisfies the property (∗).

Lemma 6 *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 5. For each j, where $1 \leq j \leq n$, do the following: if $X_j = true$, then set $\mathsf{copies}(u_j) = 1 + c_j$, where c_j is the number of clauses in which X_j is present. Else set $\mathsf{copies}(u_j) = 1$. 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 $\mathcal{A}$-complete 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 more than one copies (since the number of copies was based on a satisfying assignment for 1-in-3 SAT). If $\mathsf{copies}(u_{j_k}) > 1$, then a^i_k 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 $X_j = true$ iff $\mathsf{copies}(u_j)$ 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 $\mathcal{I}$. □

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* $\sqrt{|\mathcal{A}|}/2$.

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 $\mathsf{copies}(b)$ (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 $k \leq \mathsf{copies}(b)$ copies of b get 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 Sect. 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 $O(m(n_1 + \sqrt{C}))$ 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 $\mathsf{copies}(b)$ number of copies and call them $b_1, \ldots, b_{\mathsf{copies}(b)}$, where each b_i has the same neighborhood as the original vertex b. However, the number of vertices and edges in such a graph will be a function of the sum of number of copies of every item and therefore can be considerably larger than the number of vertices and edges in G. 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 $\mathsf{copies}(b)$ many pairs (a_i, b). Such matchings are called b-matchings in the literature; we abuse notation for the sake of convenience. It is easy to see that the structural characterization for popular matchings from Abraham et al. (2007) 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 having $\mathsf{copies}(b)$ number of occurrences of b, for every item b, 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 in the next section 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.

5.1.1 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 the 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 $\mathsf{copies}(b)$ 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 $\mathsf{copies}(b)$ 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 time proportional to the number of edges in the graph provided we create $\mathsf{copies}(b)$ many occurrences of each item p and replicate the neighborhood of b for each copy of b. However this is too expensive. The main point to note is that all the $\mathsf{copies}(b)$ 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 the people that are unmatched in M_0 and all the items that are not fully matched by M_0 (i.e., an item b that is matched to fewer than $\mathsf{copies}(b)$ many people). This is because if we would have had $\mathsf{copies}(b)$ many occurrences of b, some of these occurrences would have remained unmatched by M_0 and the other occurrences 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 matched 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 an unmatched person or 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. However, we have choice in selecting items of $\mathcal{E}$ and their number of copies that should participate in the min-cost popular matching. We make this choice in the second stage of our algorithm, as described in the next section.

5.1.2 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 an 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 + $\mathsf{cost}(b)$. 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 and 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".

We present our entire algorithm in Algorithm 2. Before we prove the correctness, we use an illustrative example to demonstrate the execution of our algorithm.

Illustrative example Let $G = (\mathcal{A} \cup \mathcal{B}, E)$ denote the instance where $\mathcal{A} = \{a_1, \ldots, a_6\}$ and $\mathcal{B} = \{b_1, \ldots, b_5\}$. The preference lists of the people are as shown in Fig. 5(a), whereas for every $b \in \mathcal{B}$, $\mathsf{copies}(b)$ and $\mathsf{cost}(b)$ are as shown in Fig. 5(b). The items

Algorithm 2 Algorithm for min-cost popular matching

Construct the graph $G_1 = (\mathcal{A} \cup \mathcal{B}, E_1)$ where $E_1 = \{(a, b) : a \in \mathcal{A}, b \in f(a)\}$.
Construct the flow graph $F(G_1)$ by adding two vertices s and t and adding directed edges with appropriate capacities.
Compute a maximum flow in $F(G_1)$ and translate the flow to a matching M_0 in G_1.
Mark the vertices of G as *odd*, *even*, and *unreachable* (their status in G_1) using M_0.
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.
Delete from G' all edges between two *odd* vertices and all edges between an *odd* vertex and an *unreachable* vertex.
Delete from M_0 all edges that are incident on *odd* people in G' and call the resulting matching M_1.
$i = 1$.
while there exists an unmatched person a in M_i **do**
  Build a Hungarian tree T_a rooted at a.
  if there exists no augmenting path starting at a **then**
    Quit and declare "G does not admit any popular matching".
  else
    Augment M_i along the cheapest augmenting path in T_a and call the new matching M_{i+1}.
  end if
  $i = i + 1$.
end while
Return M_i.

a_1	$\mathbf{b_1}$	b_4	$(\underline{b_2}, \underline{b_5})$
a_2	$\mathbf{b_1}$	$\underline{b_5}$	
a_3	$(\mathbf{b_1}, \underline{\mathbf{b_2}})$	b_3	
a_4	$(\underline{\mathbf{b_2}}, \mathbf{b_3})$	b_1	
a_5	$(\underline{\mathbf{b_2}}, \mathbf{b_4})$	b_3	
a_6	$\mathbf{b_4}$	b_1	$\underline{b_5}$

(a)

	copies	*cost*
b_1 :	1	8
b_2 :	4	3
b_3 :	2	4
b_4 :	1	2
b_5 :	1	4

(b)

Fig. 5 Example to illustrate Algorithm 2

which are bold in Fig. 5(a) denote the f-items for a person, whereas items which are underlined denote the s-items for a person.

In the first stage, our algorithm constructs the graph G_1 on rank-1 edges of G and the flow-graph $F(G_1)$. It is easy to see that in this example the maximum flow in $F(G_1)$ is of 5 units. Figure 6 shows the flow graph $F(G_1)$ and a matching $M_0 = \{(a_1, b_1), (a_3, b_2), (a_4, b_2), (a_5, b_2), (a_6, b_4)\}$ in $F(G_1)$. Using this matching and building Hungarian trees rooted at unmatched people and items which are

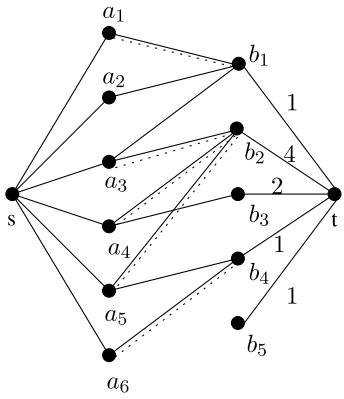

Fig. 6 The flow graph $F(G_1)$. Dotted edges indicate the matching M_0

not fully matched, we can get the partition of vertices $\mathcal{A} \cup \mathcal{B} = \mathcal{O} \cup \mathcal{E} \cup \mathcal{U}$. Here, $\mathcal{O} = \{a_3, a_4, a_5, b_1\}$, $\mathcal{E} = \{a_1, a_2, b_2, b_3\}$ and $\mathcal{U} = \{a_6, b_4\}$. At this point, we have completed Stage 1 of our algorithm or in other words, Steps 1–4 of Algorithm 2. We now identify s-items for every person using the partition of vertices and construct the graph G' as described in Step 5 of the algorithm. The s-items for every person are the underlined items in Fig. 5(a). We note that the edges (a_3, b_1) and (a_5, b_4) get deleted from the graph G' since they are $\mathcal{OO}$ and $\mathcal{OU}$ edges respectively. Finally, we delete the edges (a_3, b_2), (a_4, b_2) and (a_5, b_2) from M_0 and begin with the matching $M_1 = \{(a_1, b_1), (a_6, b_4)\}$. The above 3 edges get deleted since they are incident on vertices belonging to the set $\mathcal{O}$. Finally constructing Hungarian trees at unmatched people it is easy to see that the instance admits a popular matching and the min-cost popular matching has cost equal to 20. One such matching is $M = \{(a_1, b_2), (a_2, b_1), (a_3, b_2), (a_4, b_3), (a_5, b_2), (a_6, b_4)\}$.

Correctness of this algorithm 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 $\mathsf{copies}(b)$ many occurrences 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 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 $\mathsf{cost}(\beta_2) \leq \mathsf{cost}(\beta_1)$.

Repeating the same argument for the β_2–β_M sub-path we get vertices $\beta_3, \ldots, \beta_t = \beta_M$ where $\mathsf{cost}(\beta_2) \geq \mathsf{cost}(\beta_3) \geq \cdots \geq \mathsf{cost}(\beta_t)$. 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 the Ford-Fulkerson max-flow algorithm. Gabow's algorithm runs in time $O(\sqrt{C}m)$ where $C = \sum_{i=1}^{|\mathcal{B}|} \mathsf{copies}(b_i)$ 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 the set of 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 minimum cost.*

Note that by assigning a huge cost $\hat{C} > \sum_b \mathsf{copies}(b) \cdot \mathsf{cost}(b)$ to each of the last items ℓ_a that we introduced, where $a \in \mathcal{A}$, 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.

6 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.

It may be interesting to consider the *augmentation* model with costs in the context of other notions of optimality like rank-maximality (Irving et al. 2006) or fairness. Although rank-maximal/fair matchings are guaranteed to exist in an instance, it may be of practical interest to allocate a budget and augment the graph within the budget constraints to obtain an instance that admits the *best* rank-maximal/fair matching.

References

Abraham DJ, Cechlárová K, Manlove DF, Mehlhorn K (2004) Pareto-optimality in house allocation problems. In: Proceedings of 15th annual international symposium on algorithms and computation, pp 3–15

Abraham DJ, Irving RW, Kavitha T, Mehlhorn K (2007) Popular matchings. SIAM J Comput 37(4):1030–1045

Gärdenfors P (1975) Match making: assignments based on bilateral preferences. Behav Sci 20:166–173

Irving RW, Kavitha T, Mehlhorn K, Michail D, Paluch K (2006) Rank-maximal matchings. ACM Trans Algorithms 2(4):602–610

Kavitha T, Nasre M (2009) Note: optimal popular matchings. Discrete Appl Math 157(14):3181–3186

Kavitha T, Nasre M (2011) Popular matchings with variable item copies. Theor Comput Sci 412(12–14):1263–1274

Kavitha T, Nasre M, Nimbhorkar P (2010) Popularity at minimum cost. In: Proceedings of 21st annual international symposium on algorithms and computation, pp 145–156

Kavitha T, Mestre J, Nasre M (2011) Popular mixed matchings. Theor Comput Sci 412(24):2679–2690

Mahdian M (2006) Random popular matchings. In: Proceedings of the 8th ACM conference on electronic commerce, pp 238–242

Manlove D, Sng C (2006) Popular matchings in the capacitated house allocation problem. In: Proceedings of the 14th annual European symposium on algorithms, pp 492–503

McCutchen RM (2008) 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, pp 593–604

McDermid E, Irving RW (2011) Popular matchings: structure and algorithms. J Comb Optim 22(3):339–358

Mestre J (2008) Weighted popular matchings. In: Kao M-Y (ed) Encyclopedia of algorithms. Springer, Berlin

Pulleyblank WR (1995) Matchings and extensions. In: Handbook of combinatorics (vol. 1). MIT Press, Cambridge, pp 179–232 (Chap. 3)

Schaefer T (1978) The complexity of satisfiability problems. In: Proceedings of the 10th annual ACM symposium on theory of computing, pp 216–226