

New Models for Two Variants of Popular Matching

Danuta Sorina Chisca, Mohamed Siala, and Barry O’Sullivan
Insight Centre for Data Analytics
Department of Computer Science,
University College Cork, Ireland
{sorina.chisca|mohamed.siala|barry.osullivan}@insight-centre.org

Gilles Simonin
TASC - Institut Mines Telecom Atlantique
LS2N UMR 6004
4 rue Alfred Kastler, 44307 Nantes, France
gilles.simonin@imt-atlantique.fr

Abstract—We study the problem of matching a set of applicants to a set of posts, where each applicant has an ordinal preference list, which may contain ties, ranking a subset of posts. A matching M is *popular* if there exists no matching M' where more applicants prefer M' to M . Several notions of optimality are studied in the literature for the case of strictly ordered preference lists. In this paper we address the case involving ties and propose novel algorithmic and complexity results for this variant. Next, we focus on the NP-hard case where additional copies of posts can be added in the preference lists, called *Popular Matching with Copies*. We define new dominance rules for this problem and present several novel graph properties characterising the posts that should be copied with priority. We present a comprehensive set of experiments for the popular matching problem with copies to evaluate our dominance rules as well as the different branching strategies. Our experimental study emphasizes the importance of the dominance rules and characterises the key aspects of a good branching strategy.

I. INTRODUCTION

Matching under preferences is a large family of problems occurring in real-world applications, including campus house allocation, exchanges/markets, and assigning reviewers to conference papers [1]. Different formulations of these problems have been proposed, distinguishing between *one-sided* matching [2] and *two-sided* matching, e.g. the stable marriage problem [3], [4]. The notion of *popularity* was introduced by Gardenfors [5] in the full stable marriage problem, where it is a desirable property in finding a weakly stable matching.

In the *popular matching* problem, a set of applicants \mathcal{A} have to be matched to a set of posts \mathcal{P} whereby each applicant has an ordinal list of preferences over posts, ranking a subset of posts in order of preference. Notice that only the applicants have preferences over posts. Therefore, the popular matching problem considers only one-sided preferences. If applicants can be indifferent between posts, we say that preference lists contain ties. A formal definition of popular matching is given later in Section II. Informally, a matching M can be seen as a set containing pairs $\langle a, p \rangle$ where a is an applicant and p is a post and each applicant/post appears at most once in M . Given two matchings M and M' , we use the notation $M \prec M'$ if (strictly) more applicants prefer M' to M . A matching M is *popular if and only if* there is no matching M' such that $M \prec M'$. Consider the following example from [6].

Example 1.1: Let $\mathcal{A} = \{a_1, a_2, a_3\}$, $\mathcal{P} = \{p_1, p_2, p_3\}$ and each applicant prefers p_1 to p_2 and p_2 to p_3 . The reader can check that this instance does not admit a popular

matching since for any matching M , there exists a matching M' such that $M \prec M'$. For instance, consider the three (symmetrical) matchings $M_1 = \{(a_1, p_1), (a_2, p_2), (a_3, p_3)\}$, $M_2 = \{(a_1, p_3), (a_2, p_1), (a_3, p_2)\}$, and $M_3 = \{(a_1, p_2), (a_2, p_3), (a_3, p_1)\}$. We have $M_1 \prec M_2$, $M_2 \prec M_3$, and $M_3 \prec M_1$.

A solution to this situation is to add a copy of a post p_1 or p_2 . This problem is called the *Popular Matching with Post Copies*. An example for the popular matching problem is the training program. A training program can be run for a single person, but some training programs may be able to accommodate more than one person. We wish to fix the capacity of each training program so as to enable the resulting instance to admit a popular matching. Another relevant scenario is to organise bus tours at a conference. People can express their preference over the tours and it is possible to increase the size of a bus at a cost.

Related previous work. Abraham et al. in [6] showed that the standard popular matching problem is polynomial to solve even if the preference lists contain ties. In particular, they describe a number of efficient algorithms for finding popular matchings. Recently, it has been shown that this problem can be encoded using one Global Cardinality Constraint [7]. Several variants of the popular matching problem exist in the literature. Let r be the largest preference list size. A popular matching is *fair* [8] if it matches the least number of applicants to their r -th choice and then, subject to this, the least number to their $(r - 1)$ -th choice, and so on. A matching is *rank-maximal* [9] if it matches the maximum number of applicants to their first choice and then, subject to this, the maximum number to their second choice, and so on. The authors of [10], [11] studied the problem of finding a popular matching in a setting where additional copies of posts can be introduced at a cost. A variant called popular matching in the capacitated house allocation problem is described in [12] and its variant with weights is introduced in [13].

Contributions. In this paper we study two important variants of the popular matching problem. The first variant is the optimal popular matching. The notion of optimality is studied in the literature solely for the case of preference lists without ties. In this paper we extend to the case involving ties and propose novel algorithmic and complexity results for this variant. In particular, we propose an integer linear

programming (ILP) model and a flow model for the optimal popular matching, where ties are allowed in the preference list. We show that the optimal popular matching with ties can be solved in $O(n(m + n \log(n)))$ time, where n is the set of applicants and m is the length of the preference lists. The second variant deals with instances that does not admit a popular matching (Example 1.1). We consider the case where we can add copies of posts in order to find a popular matching. This problem is known to be NP-Hard [10] and it remains NP-complete even when preferences are derived from the same master preference list [14]. Kavitha et al. in [10] show that this problem remains NP-complete when the preference lists are of length 2 and ties are allowed only in first position. However, their properties are not applicable to the general case. We develop a number of new graph properties that are used to define new dominance rules for the popular matching with copies problem. These dominance rules have a significant impact for the efficiency of the branching strategy. We conduct the first experimental study for this variant by evaluating the different search strategies as well as the dominance rules that we propose. Our experiments show the key aspects of a good branching strategy and demonstrate the efficiency of our dominance rules.

The remainder of the paper is organized as follows. In Section II we give the formal background and define the notation used in the paper. In Section III we give our algorithmic results on optimal popular matching. In Section IV we study popular matching with copies, presenting novel graph properties and new dominance rules. Lastly, in Section V we present our experimental study.

II. POPULAR MATCHING

An instance of the *popular matching* problem involves an undirected bipartite graph $G = (A \cup P, E)$, where A is called the set of applicants, P is called the set of posts, and each applicant a_i ranks all posts in $\{p_j \mid (a_i, p_j) \in E\}$. We assume that every applicant $a_i \in A$ has in its preference list an extra unique post l_i , called a last resort. Every last resort l_i is assumed to be worse than any post in P . In this way, we can assume that every applicant is matched, since any unmatched applicant can be assigned to his unique last resort.

For each matched node u in a matching M , we denote by $M(u)$ the node linked to u by an edge in M . An applicant a_i prefers a matching M to a matching M' if a_i is matched in M and unmatched in M' , or if a_i is matched in both M and M' but prefers $M(a_i)$ to $M'(a_i)$. A matching M is *more popular* than a matching M' if there are more applicants preferring M to M' . A matching M is *popular* if there exists no matching more popular than M . The popular matching problem is the question of deciding if a popular matching exists, and to find one if it is the case.

A. Strict Preference Lists

Suppose that the preferences are given in a strict order (i.e., without ties). For each applicant a_i , we denote by $f(a_i)$ the first-ranked post in its preference list. A post $p_j \in P$ is called

a_1 :	P1	p_2	p_3	$\underline{l_1}$
a_2 :	P1	$\underline{p_5}$	p_4	$\underline{l_2}$
a_3 :	P2	p_1	p_3	$\underline{l_3}$
a_4 :	P2	p_3	$\underline{p_6}$	$\underline{l_4}$
a_5 :	P2	$\underline{p_6}$	p_4	$\underline{l_5}$
a_6 :	P3	p_2	$\underline{p_5}$	$\underline{l_6}$

Fig. 1. An instance of popular matching with strict preference lists [6].

an *f-post* if $\exists a_i \in A$ such that $f(a_i) = p_j$. Let $f(p_j)$ be the set of applicants a_i where $f(a_i) = p_j$. Let $s(a_i)$ be the best (most preferred) post for a_i that is not an *f-post*. Such a post is called an *s-post*. The least preferred post l_i guarantees the existence of $s(a_i)$. Figure 1 shows an example with 6 applicants and 6 posts [6]. The bold entries in the preference lists are the *f-posts* and the underlined entries are the *s-posts*.

Theorem 2.1: (From [6]) A matching M is popular if and only if:

- Every *f-post* is matched in M ,
- For each applicant a_i , $M(a_i) \in \{f(a_i), s(a_i)\}$.

B. Preference List With Ties

We consider the case where preferences can contain ties (see Figure 2(a) from [6]). The notions of *f-posts* and *s-posts* are defined in the case where ties are present, similar to preference lists without ties.

The definition of $f(a_i)$ becomes the set of top choices for applicant a_i . However the definition of $s(a_i)$ is no longer the same and, in this case, it may contain a number of surplus *f-posts*. Let M be a popular matching of some instance graph $G = (A \cup P, E)$. We define the *first-rank* graph of G as $G_1 = (A \cup P, E_1)$, where E_1 is the set of all rank-one edges.

Let M_1 be a maximum matching in G_1 . We recall the Gallai-Edmonds decomposition [15] in Lemma 2.1 using the following notation. Using M_1 , one can partition $A \cup P$ into three disjoint sets: \mathcal{E} , \mathcal{O} , and \mathcal{U} . These sets are defined as follows: \mathcal{E} (respectively \mathcal{O}) is the set of nodes having an even (respectively odd) alternating path with respect to M_1 from an unmatched node; and \mathcal{U} is the set of nodes that are not in $\mathcal{E} \cup \mathcal{O}$.

Lemma 2.1 (Gallai-Edmonds decomposition (From [15])):

Let \mathcal{E} , \mathcal{O} and \mathcal{U} be the vertices sets defined by G_1 and M_1 above. Then:

- \mathcal{E} , \mathcal{O} , and \mathcal{U} are a partition of $A \cup P$, and any maximum matching in G_1 leads to exactly the same sets \mathcal{E} , \mathcal{O} , \mathcal{U} .
- Every node in \mathcal{O} (resp. \mathcal{U}) is matched to a node in \mathcal{E} (resp. \mathcal{U}), and $|M| = |\mathcal{O}| + |\mathcal{U}|/2$.
- No maximum matching of G_1 contains an edge between two nodes in \mathcal{O} , a node in \mathcal{O} and a node in \mathcal{U} , or between a node in \mathcal{E} and a node in \mathcal{U} .

In [6], $s(a_i)$ is defined as the top choice(s) for a_i in \mathcal{E} ; see Figures 2(a) where bold entries represent the *f-posts* and underlined entries represent the *s-posts*. In this example $\mathcal{E} = \{a_1, a_2, a_3, a_4, p_3, p_5, p_6, l_1, l_2, l_3, l_4, l_5, l_6\}$, $\mathcal{O} = \{a_6, p_1, p_2\}$, and $\mathcal{U} = \{a_5, p_4\}$.

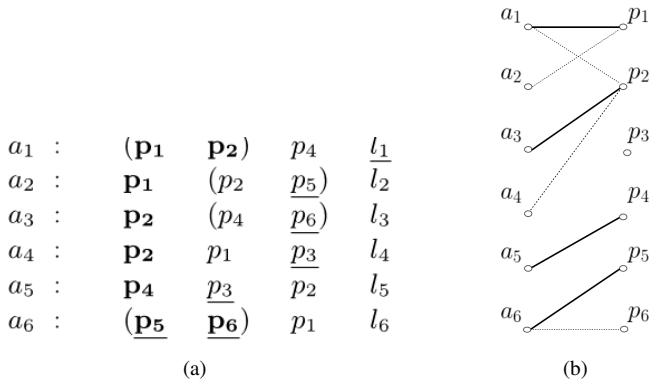


Fig. 2. An example with ties in the preference lists and the graph G_1 with a maximum matching in bold.

Theorem 2.2: (From [6]) A matching M is popular iff:

- $M \cap E_1$ is a maximum matching of G_1 ,
- For each applicant a_i , $M(a_i) \in f(a_i) \cup s(a_i)$.

III. OPTIMAL POPULAR MATCHING

Several notions of optimality are studied in the literature for the case of preferences without ties [8], [16]. The most common way is to associate each edge with a weight. The purpose then is to find a minimum/maximum weight popular matching.

The authors of [16] improve the algorithms of Kavitha [8] for finding a min-cost popular matching in $O(n + m)$ time and rank-maximal/fair popular matchings in $O(n \log n + m)$ time, where $n = |A|$ and m is the sum of preference lists.

We consider the more general case where preferences can contain ties. We assume that the weight function is monotonically increasing with respect to the rank. In the case of maximisation, it is sufficient to consider the opposite of each weight then solve the new minimisation problem.

To the best of our knowledge, the complexity of finding an optimal popular matching where ties are allowed in the preference lists, is unknown. However, recently we found a technical report in ArXiv [17] showing the result with the same worst-case time complexity. Our approach is, however, much simpler.

We shall use a one-one correspondence between the set of posts and the set of values $\Delta = \{1, 2, \dots, |P| + n\}$ as follows: every post p_j is associated with the value j and every last resort post l_i (related to applicant a_i) is associated with the value $|P| + i$. The solution of the popular matching problem is a matching between the set of applicants A and the set of posts Δ^P corresponding to the values in Δ . Let n_1 be the number of applicants, and $n_2 \leq n_1$ be the number of posts in Δ^P . We denote by Δ_β the set of posts in $\mathcal{O}_p \cup \mathcal{U}_p$, where β is the size of this set. Let $G' = (A \cup \Delta^P, E')$ be the bipartite graph modelling our problem, where the weight on each edge $(a_i, p_j) \in E'$ represents the rank of p_j in the preference lists of a_i .

We use the well-known Minimum Weight Perfect Matching in Bipartite Graph problem. This problem is equivalent to the

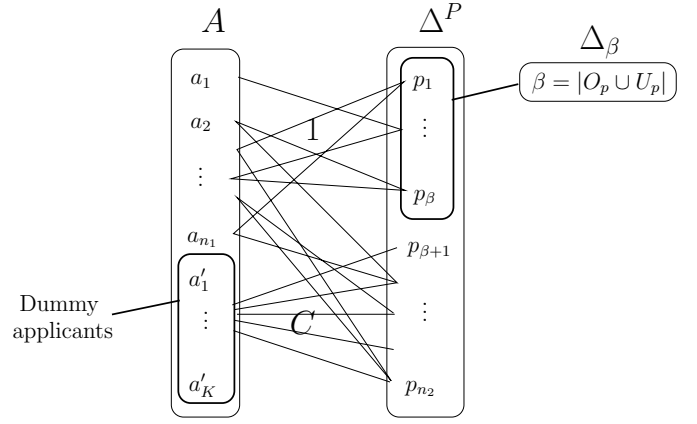


Fig. 3. A perfect matching instance.

classic assignment problem with cost, which can be solved with the following integer linear programming (ILP) model:

$$\min \sum_{(a_i, p_j) \in E'} w_i^j x_i^j \quad (1)$$

$$\sum_j x_i^j = 1 \quad \forall i \quad (2)$$

$$\sum_i x_i^j = 1 \quad \forall j \quad (3)$$

$$x_i^j = \begin{cases} 1 & \text{if the edge } (a_i, p_j) \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

We transform the graph G' to an instance of the Minimum Weight Perfect Matching in a Bipartite Graph problem. In order to obtain a perfect matching, we need to add $K = n_2 - n_1$ dummy-applicant vertices in A , each of which is linked to all posts in $\Delta^P \setminus \Delta_\beta$ with a weight equal to a large constant C , e.g. $C = |P|^2$. Note that the weight on the edges between $a_i \in A$ and $p_j \in \Delta_\beta$ is equal to 1 (first rank). This linear transformation is illustrated in Figure 3.

If we compute a Minimum Weight Perfect Matching with the ILP and find a solution, we will obtain a weight equal to $KC + \epsilon$, where ϵ represents the weights sum from the n_1 normal applicants. Since KC is constant, ϵ is the minimum weight that we are looking for. Thus, the matching obtained, without the dummy-applicant vertices, corresponds to a solution of optimal popular matching problem.

Kuhn [18] gave the first polynomial algorithm, called the Hungarian method, to solve the Minimum Weight Perfect Matching in Bipartite Graph problem. The algorithm had a running time of $O(n^3)$. Then, in [19] the authors gave a reduction of this problem to the flow circulation problem. As we did with our linear transformation to obtain a perfect matching instance, we can create a flow circulation instance in a special network. And from this new model, we can use better existing complexity results.

Let $G = (V, E)$ be a network, the flow circulation problem

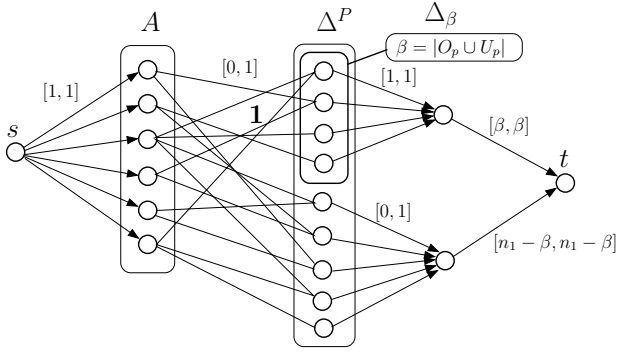


Fig. 4. Illustration of the flow circulation problem.

is defined as follows:

- $lb(v, w)$ lower bound on flow on the edge (v, w) ,
- $ub(v, w)$ upper bound on flow on the edge (v, w) ,
- $c(v, w)$ cost of one flow unit on the edge (v, w) ,

$$lb(v, w) \leq f(v, w) \leq ub(v, w), \quad (4)$$

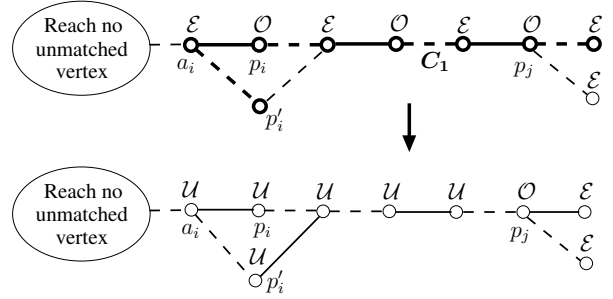
$$\min \sum_{(v, w) \in E} c(v, w) \cdot f(v, w). \quad (5)$$

Linear transformation. In order to force the assignment of the β posts in Δ_β , we fix by $[1, 1]$ the lower and upper bounds of the arcs going out from these posts. We also force the flow passing by the other posts to not exceed $n_1 - \beta$, otherwise all the flow could pass by these arcs and not by the posts in Δ_β . An illustration of the construction is presented in Figure 4. In [20], the authors proposed an algorithm to solve the flow problem in $O(n(m + n \log n))$. Therefore, an optimal popular matching with ties can be computed in $O(n(m + n \log n))$ time.

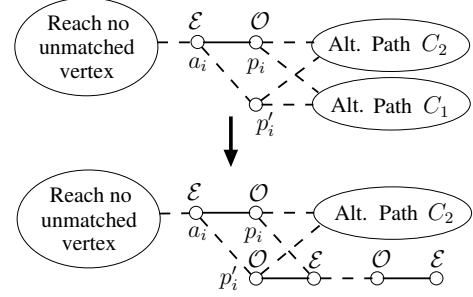
IV. POPULAR MATCHING WITH POST COPIES

Many instances of the popular matching problem admit no solution. In the literature authors have studied such situations where extra copies of posts are allowed in order to find a solution [10]. In [10] the authors show that this problem remains NP-complete even when the preference lists have length 2 and ties are allowed only in first position. They also consider the variant where they maintain an upper bound k on the total number of extra copies for all posts, rather than an upper bound on the number of copies for each item.

The problem is related to many real-world applications such as DVD rental stores, training programs, and bus tours. According to client demand, additional copies of an item can be purchased in order to satisfy their preferences. A training program can be run for a single person, but some training programs may be able to accommodate more than one person. We wish to fix the capacity of each training program. To organise bus tours for a given conference, people express their preference over the tours and it is possible to increase the size of a bus at a cost.



(a) Non-disjoint alternating paths



(b) Disjoint alternating paths

Fig. 5. Illustrations of the copy of a post in \mathcal{O} in G_1 .

This problem is called the **FIXINGCOPIES** problem:

Instance: Given an instance of the popular matching problem associated to the graph $G = (A \cup P, E)$, and a list $\langle c_1, \dots, c_{|P|} \rangle$ of upper bounds on the number of copies possible for each post.

Question: Does there exist an $\langle \chi_1, \dots, \chi_{|P|} \rangle$ such that for each $i \in \{1, \dots, |P|\}$ having χ_i copies of the i -th post, where $1 \leq \chi_i \leq c_i$, enables the resulting graph to admit a popular matching?

The **FIXINGCOPIES** problem is known to be NP-complete [10], and it remains NP-complete even when preferences are derived from the same master preference list [14].

A. Properties of the **FIXINGCOPIES** Problem

We assume that the initial instance \mathcal{I}^0 does not admit a popular matching. The bipartite graph associated to \mathcal{I}^0 is denoted by $G^0 = (A \cup P^0, E^0)$.

FIXINGCOPIES can be divided into two parts: first, deciding the number of copies for each post; second, solving the new popular matching instance. We propose to study the consequences of creating post copies for instances that do not admit popular matching according to the labels.

1) *An Automaton for the Posts:* We introduce several properties of the new instances obtained by copying a post according to the $\mathcal{E}, \mathcal{O}, \mathcal{U}$ labelling. In the following, a post will be said to be in \mathcal{E}, \mathcal{O} or \mathcal{U} if it is labelled by the corresponding set.

Lemma 4.1: In any instance obtained from the copy of a post p_i in \mathcal{O} , the post and its copies will remain in \mathcal{O} or be in

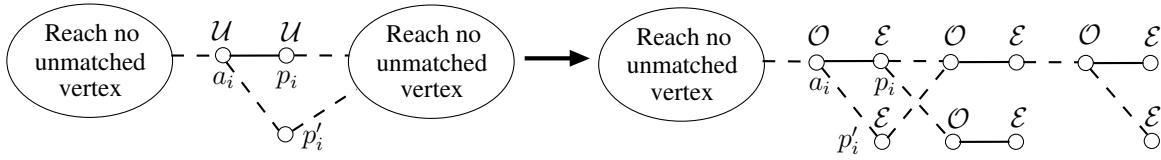


Fig. 6. Illustration of the copy of a post in \mathcal{U} in G_1 .

\mathcal{U} . From this copy, any other post in \mathcal{O} will be also in \mathcal{U} or will remain in \mathcal{O} , and all posts in \mathcal{E} or \mathcal{U} will stay the same.

Proof. Let M be the maximum matching used to obtain the sets \mathcal{E} , \mathcal{O} and \mathcal{U} . We use the M -alternating path notion from Berge [21]. By definition, if p_i is in \mathcal{O} then it is linked to an M -alternating path and matched to an applicant a_i in \mathcal{E} (see Figure 5). We recall that all posts in these paths are in \mathcal{O} by definition. Therefore there are two cases: 1) there are only non-disjoint M -alternating paths linked to p_i (see Figure 5(a)), and 2) at least two disjoint M -alternating paths are connected to p_i (see Figure 5(b)).

In both cases, the copy p'_i is also connected to a_i and to at least one M -alternating path. There exists one augmenting M -alternating path called C_1 from p'_i . By changing the membership in M of all edges of C_1 , we can increase the matching. After this modification, there is no other augmenting M -alternating path and the matching is maximum by Berge's Theorem [21]. Since all posts of C_1 are in \mathcal{O} , no post in \mathcal{E} or \mathcal{U} changes from this copy. From this new matching M , we can recompute the labels.

In the first case, let p_j be the closest vertex to p_i on C_1 which creates a joint with other M -alternating paths. To simplify the proof we split C_1 in two parts: the one at the left of p_j (containing p_i) and the right one. The right part of C_1 remains unchanged, but on the left part of C_1 , every vertex changes into \mathcal{U} (see Figure 5(a)). Thus from this copy, all posts in \mathcal{O} will remain in \mathcal{O} or will change into \mathcal{U} .

In the second case, there exists at least one distinct M -alternating path C_2 connected to p_i and p'_i . The concatenation of C_2 and the modified C_1 creates a unique M -alternating path. Therefore, the labels from the modified C_1 remain the same. Thus in both cases, after the copy of a post in \mathcal{O} , all posts in \mathcal{O} will remain the same or will be in \mathcal{U} , and all posts in \mathcal{E} or \mathcal{U} will remain the same. \square

Corollary 4.1: In any instance obtained from a copy of a post in \mathcal{O} , the maximum matching is increased by 1.

Lemma 4.2: In any instance obtained from the copy of a post p_i in \mathcal{U} , the post and its copies will be in \mathcal{E} . From this copy, any other post in \mathcal{U} will be in \mathcal{E} or will remain in \mathcal{U} . And all posts in \mathcal{E} or \mathcal{O} will remain the same.

Proof. Let M be the maximum matching used to obtain the sets \mathcal{E} , \mathcal{O} and \mathcal{U} . By definition, if p_i is in \mathcal{U} then it cannot be linked to an M -alternating path, and it is matched to an applicant a_i who is also in \mathcal{U} (see Figure 6). Since the copy p'_i has the same properties than p_i , no augmenting M -alternating path is created from it, and p'_i remains unmatched and in \mathcal{E} . Thus the maximum matching remains the same, so all vertices

that are not a member of any M -alternating path from p'_i will keep their label.

All the vertices in \mathcal{U} that are a member of an M -alternating path from this p'_i will be in \mathcal{O} or \mathcal{E} . It is obvious that the applicant vertices will be in \mathcal{O} and the post ones in \mathcal{E} . All vertices not in \mathcal{U} that are also a member of this path will remain the same. Thus after the copy of a post in \mathcal{U} , all posts in \mathcal{U} will remain the same or will be in \mathcal{E} , and all posts in \mathcal{E} or \mathcal{O} will remain the same. \square

Corollary 4.2: In any instance obtained from the copy of a post in \mathcal{U} , the maximum matching will remain the same.

Corollary 4.3: From any instance \mathcal{I} with a popular matching, every instance \mathcal{I}^* obtained from \mathcal{I} by copying only posts in \mathcal{E} admits a popular matching

Proof. From Lemma 4.3, adding more copies from \mathcal{E} will not change the labels. In the best case we will obtain more s -posts. Thus \mathcal{I} is a sub-solution of any \mathcal{I}^* . \square

Lemma 4.3: In any instance obtained from the copy of a post p_i in \mathcal{E} , the maximum matching and the label of any vertex will remain the same.

Proof. Let M be the maximum matching used to obtain the sets \mathcal{E} , \mathcal{O} , and \mathcal{U} . First, if p_i is unmatched then any copy of this post will be in \mathcal{E} . Second, if p_i is matched, then by definition it cannot be linked to an M -alternating path, and it is matched to an applicant a_i who is in \mathcal{O} (see Figure 7). Obviously, p'_i will be only connected to an applicant vertex that is in \mathcal{O} . Thus, we will have only new M -alternating paths from p'_i but none of them will be an augmenting path. Thus after the copy of a post in \mathcal{E} , all vertices will remain the same as the cardinality of the maximum matching. \square

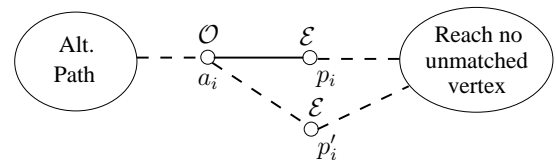


Fig. 7. Illustration of the copy of a post in \mathcal{E} in G_1 .

From Lemmas 4.1, 4.2 and 4.3 we can deduce an important result about the label behaviour of posts after a copy.

Theorem 4.1: From any copy of a post in \mathcal{E} , \mathcal{O} or \mathcal{U} , every post follows the automaton shown in Figure 8 (to characterise its new label).

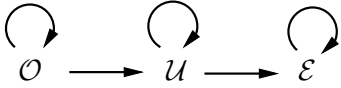


Fig. 8. Illustration of the automaton

2) *Dominance rules:* We introduce new dominance rules. For any instance \mathcal{I} , we denote by $\mathcal{F}^{\mathcal{I}}$ the set of posts that are not f -posts or s -posts.

Lemma 4.4: Let \mathcal{I} and \mathcal{I}^* be two instances where \mathcal{I}^* is built from \mathcal{I} by copying some posts. Any post in $\mathcal{F}^{\mathcal{I}}$ is in $\mathcal{F}^{\mathcal{I}^*}$.

Proof. Let p_i be such a post. Clearly p_i is in \mathcal{E} . Now since $p_i \in \mathcal{F}^{\mathcal{I}}$, then for all the applicants, their best choice in \mathcal{E} is never p_i but other posts in \mathcal{E} . From Theorem 4.1 any post in \mathcal{E} will always remain in \mathcal{E} after any kind of copy. Thus, the other posts will remain in \mathcal{E} and p_i will never become a s -post. \square

Theorem 4.2: Let \mathcal{I} be an instance without a popular matching, and let $p_i \in \mathcal{F}^{\mathcal{I}}$. Every instance with a popular matching, obtained from \mathcal{I} by copying some posts, admits a popular matching even with the original number of copies of p_i from \mathcal{I} .

Proof. If $p_i \in \mathcal{F}^{\mathcal{I}}$, it cannot be assigned to an applicant in a popular matching. By Lemma 4.4, the proof is immediate. \square

The first dominance rule works for any copy of any post. Using the Corollary 4.3 another dominance rule can be described for the posts in \mathcal{E} .

We say that two instances \mathcal{I} and \mathcal{I}' have the same status if \mathcal{I} admits a popular matching iff \mathcal{I}' admits a popular matching.

Theorem 4.3: Every instance obtained from an instance \mathcal{I} by copying a post p_i in \mathcal{E} has the same status as \mathcal{I} or admits a popular matching.

Proof. By using Lemma 4.3 and Corollary 4.3, one can conclude easily that any copy of a post in \mathcal{E} will not change the labels, and may add more s -posts in the instance. This can lead to finding a popular matching. \square

Remark 4.1: These strong theoretical results consider only instances with a maximum number of copies for the s -posts and to avoid the posts in $\mathcal{F}^{\mathcal{I}}$. These properties highlight the difficulty in identifying posts that do not need to be copied to find a popular matching. Indeed, the s -posts can change only after a copy of \mathcal{U} posts.

B. Modelling the FIXINGCOPIES Problem

We propose to explore a lexicographic instance tree where the root node represents \mathcal{I}^0 and any other node is associated with an instance obtained from its parent by increasing the copies of one post. With lexicographic branching, we explore the instance nodes where we increase the number of copies of each post to the maximum before increasing the next ones.

Let $\chi^k(p_i)$ be the number of copies fixed for a post p_i given an instance \mathcal{I}^k . In this tree, any descendant \mathcal{I}^l of a node \mathcal{I}^k has the following property: $\forall p_i \in P, \chi^l(p_i) \geq \chi^k(p_i)$. Thus from each node, we can apply all the results

obtained in Section IV-A. From Theorem 4.2, we can fix the number of copies of some posts for an instance node and its children. Therefore we apply the following linear time dominance rule: Let \mathcal{I}^l be a child of \mathcal{I}^k , then: $\forall i, \text{if } p_i \in \mathcal{F}^{\mathcal{I}^k}$, then $\forall \mathcal{I}^l, \chi^l(p_i) = \chi^k(p_i)$.

We consider the impact of the branching strategies based on the $\mathcal{O}, \mathcal{E}, \mathcal{U}$ labelling. From Lemma 4.3 and Theorem 4.3 the copy from a post in \mathcal{E} does not change the labels and cannot remove popularity from other instances with copies from different labels. This result shows that branching on posts in \mathcal{E} does not avoid some potential solutions in another branches of the search tree. At last from Corollary 4.2 the copy from a post in \mathcal{U} does not change the maximum matching, but changes at least two posts into \mathcal{E} . This leads to new s -post assignments, and thus to remove some posts in \mathcal{E} from an s -post assignment. We can conclude that only the posts in \mathcal{U} have a significant impact on the branching decision, indeed exploring the nodes, where labels \mathcal{U} are considered last, should reduce the backtracking.

V. EMPIRICAL EVALUATION

We present an experimental study for the NP-hard variant of FIXINGCOPIES problem. Our purpose is to evaluate the different branching strategies as well as the impact of the dominance rules. The exploration strategies are the six permutations to rank in priority the three labels $\mathcal{E}, \mathcal{O}, \mathcal{U}$. The model that we presented in Section IV-B is implemented in Python. We use Numberjack [22] to model the standard popular matching problem with the CP model of [7]. The timeout is fixed to 20 minutes for every instance.

We first run all the configurations on purely random instances, but observed that these instances are easy to solve. We therefore studied another family of instances based on a notion of a master preference list. These instances are inspired by real-world situations where often the preferences follow a similar tendency. Irving et al. [14] considered the stable marriage problem in the presence of master preference lists, reflecting that the preferences of one gender are similar to each other, and proved that many interesting variants remain hard under this master list model. We adopted a similar approach in these experiments.

We generated the instances as follows: the number of applicants $a \in \{100, 150, 200\}$; the number of posts $p \in \{\frac{a}{2}, a\}$; the upper bound $ub \in \{\frac{a}{20}, \frac{a}{10}\}$ for each post copies; and each post has a probability $h \in \{50\%, 70\%, 90\%\}$ to be in the applicant preference list following the order given by the master preference list. For each configuration $\langle a, p, ub, h \rangle$ we use 40 different randomised seeds.

We report a representative set of the results in Tables I and II due to space. In these tables, every column is denoted by $X_1a_X_2p_X_3ub_X_4h$ where: X_1 is the number of applicants; X_2 is the number of posts; X_3 is the upper bound for copies; and X_4 is the probability h described above. Each row summarises the results of each branching strategy. For instance, the rows starting with eou correspond to the search strategy where every post in \mathcal{E} has the priority to be copied

