Weak Stable Matchings with Tenants and Ties

Ana Paula Tomás *

DCC-FCUP & LIACC
University of Porto, Portugal
apt@ncc.up.pt

Abstract. The paper addresses a variant of the stable marriage problem that models a job recruitment problem in which applicants are strictly ordered by priority but their preference lists may have ties. Some applicants may hold a post initially. These posts may be assigned to other applicants if their holders get another post. By reducing the problem to a sequence of maximum cardinality bipartite matching problems, combined with an effective propagation of the stability constraints, we show that applicant-optimal stable matchings may be found efficiently.

1 Introduction

Bipartite matching problems with preferences have been extensively studied in economics, operational research and computer science due to their practical applications and their mathematical structure [1, 2, 5, 9, 10, 12, 18, 20, 23]. Preferences may be expressed on both sides or one side only. The preference lists may be strictly or partially ordered and complete or incomplete, modelling situations where it is allowed or disallowed to express indifference and inadmissible pairs. Problems with bilateral preferences have been extensively studied and are usually modelled by variants of the classical *stable marriage problem* [5, 9, 15]. Problems with unilateral preferences, namely *popular* matchings, first considered by Gardenfors [6] and *rank-optimal* matchings, introduced by Irving [11], have received much attention recently [2, 12, 18, 20].

In the context of job assignment problems, a matching M is called popular if there is no matching M' such that the applicants preferring M' to M outnumber the applicants preferring M to M'. A rank-maximal (or greedy) matching is one in which the maximum possible number of applicants are matched to their first choice post, and subject to that condition, the maximum possible number are matched to their second choice post, and so forth.

* Work partially funded by LIACC through Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia (FCT) and Programa POSI.

A matching M is *GS-stable* (more often, *weak-stable* [5, 9, 17]) iff there is no pair (a, p) such that applicant a strictly prefers post p to the post he got matched to in M and p is unassigned or strictly prefers a to the applicant assigned to it in M.

In this paper, we address a job assignment problem in which applicants are strictly ranked by priority, their preference lists for posts may be incomplete and may contain ties and some applicants may hold posts initially. These posts may be assigned to other applicants only if they get a new post. The goal is to find *an applicant-optimal stable matching*, that is, a matching that assigns the best post to each applicant and does not violate their relative priorities, which ultimately define who is matched first. We call it *teachers recruitment* (TRecruitment, for short), for its origin was a real-life teachers recruitment problem.

Actually, this work was motivated by a great controversy involving teachers recruitment for the portuguese state schools in 2004 [24]. The recruitment program has a national scope, with more than one hundred thousand applicants but comparatively few posts. Applicants are strictly ranked and those that have already tenured positions cannot result unmatched. In the worst case, they will keep their current positions. Applicants indicate their strict preferences for schools or/and regions (but cannot list more than 100 schools, 50 geographic regions and 23 pedagogic regions, the latter covering all the state schools), and also for four different week lecture loads. They may interleave schools and regions, as they wish, indicating a region if they have equal preference for all the schools in that region.

In part, the controversy arose because the published results violated the relative priorities of candidates, that is, the matchings were unstable. In the end, a new algorithm for finding a stable matching was explained in the media. It had worst time-complexity $O(\max(t, 1)nk)$, for k posts and n applicants, t having tenured positions or, more accurately, $O(\max(t, 1)m)$, where m is the total length of the preference lists. It assumed that the preference lists contained no ties. For breaking ties, schools within regions were strictly ordered by a school code. This rule was not consistently stated in all the official regulations. However, it is not hard to imagine scenarios where some applicants result unfairly unmatched just because schools were sorted for breaking ties in some fashion instead of another.

Hence, on the one hand, our work results from the intuition that better algorithms could be achieved by exploiting the relation of TRecruitment to the stable marriage problem. On the other hand, we wondered whether applicant-optimal stable matchings could be found efficiently if

ties were not broken. Indeed, TRecruitment is related to Stable Marriage with Ties and Incomplete Lists and it is known that, for the latter, some problems may be hard and hard to approximate [13, 17], e.g., determining whether there is a stable matching that contains a given pair.

To model TRecruitment as a variant of Stable Marriage we need to consider that each applicant who had already a permanent post is top rank for his initial post. Thus, there is not a unique preference list for all posts, that is, somehow preferences are expressed on both sides.

We show that, when the preference lists of applicants are strictly ordered, TRecruitment may be solved in $O(m)$ using the Gale-Shapley Algorithm (applicant-oriented) [5, 9]. This may be a considerable improvement when compared to the $O(\max(t, 1)m)$ algorithm used previously. We show also that in this case there is a *unique* stable matching that is admissible for TRecruitment. Moreover, we developed an efficient polynomial algorithm for solving the problem for the case when the preference lists may have ties. It is based on a reduction of TRecruitment to a sequence of maximum cardinality bipartite matching problems on reduced graphs, combined with an effective propagation of the stability constraints.

1.1 Weighted Popular Matchings and TRecruitment

The existence of tenured positions and the requirement that the matching is stable make TRecruitment distinct from the *weighted popular matchings* problem, recently studied by Mestre [20]. The latter is a variant of popular matchings where applicants have weights, a matching M being weighted popular iff there is no matching M' such that the total weight of applicants preferring M' to M exceeds the total weight of applicants preferring M to M'. Example 1 illustrates this point.

Example 1. Consider an instance of TRecruitment involving three applicants a_1, a_2 and a_3, given in decreasing priority order, and three posts p_1, p_2 and p_3, being p_1 and p_3 the initial posts of a_2 and a_3, respectively. Suppose that a_1 strictly prefers p_1 to p_2 and is not interested in p_3, a_2 only wants p_3 and a_3 only wants p_1, so that, the situation is as follows.

$$a_1 : p_1, p_2 \text{ (no post)} \qquad a_2 : p_3 \text{ (holds } p_1) \qquad a_3 : p_1 \text{ (holds } p_3)$$

The matching $M_1 = \{(a_1, p_2), (a_2, p_3), (a_3, p_1)\}$ is more popular than $M_2 = \{(a_1, p_2), (a_2, p_1), (a_3, p_3)\}$, but M_2 is the unique solution of TRecruitment. Even if it seems a nonsense to adopt M_2 instead of the

unstable matching M_1, the fact is that stability is crucial for real-life applications to avoid controversy and help applicants trust the results.

It is interesting to observe that popular matchings regained considerable interest recently [2, 18, 20]. Abraham et al. [2] gave the first polynomial-time algorithms to determine if an instance admits a popular matching, and to find a largest such matching, if one exists. For that, they developed an efficiently-checkable characterization of popular matchings, showing that the problem may be reduced to maximum matching in a restricted subgraph (defined by the so-called *first* and *second choice* jobs). The algorithm proposed by Mestre [20] for finding weighted popular matchings follows a similar approach, although he had to work out an appropriate characterization also. The existence of tenured positions in TRecruitment does not allow us to confine attention to *first* and *second* jobs, in the sense given by [20].

The rest of paper is structured as follows. In Section 2, we introduce some notation and describe the problem formally. In Section 3, we exploit the relation of TRecruitment to Stable Marriage and show that when the preference lists are strictly ordered the problem admits a unique solution. Then, in Sections 4 and 5 we consider the general case. We show that TRecruitment is polynomially solvable, also by designing an algorithm for doing it efficiently. A preliminary analysis of complexity led $O(n^2 C \max(1, t))$ time and $O(nC)$ space, where C is the maximum number of posts equally ranked by some applicant. This time bound does not take into account the effect of the propagation of the stability constraints. The algorithm has been implemented in an imperative language. Empirical results indicate that it runs quite fast and scales up efficiently.

2 The Recruitment Problem and Stable Marriage

Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be the set of applicants, strictly ordered by decreasing priority and let $\mathcal{P} = \{p_1, \ldots, p_k\}$ be the set of posts. The latter includes also the current posts of applicants, if they hold any. For each applicant a_i, let $\Gamma_{i,1}, \Gamma_{i,2}, \ldots, \Gamma_{i,\gamma_i}$ be his preference list, in strictly increasing order. Each $\Gamma_{i,l}$ is a set of posts for which a_i has the same preference, for all $1 \leq l \leq \gamma_i$. As we mentioned above, γ_i is limited by a constant, say U, that can be much smaller than k. We then append either the permanent post of a_i to that list or a fictitious post p'_i (meaning, no post), getting an *extended preference list*:

$$a_i : \begin{cases} \Gamma_{i,1}, \Gamma_{i,2}, \ldots, \Gamma_{i,\gamma_i}, \{p_j\}, & \text{if } p_j \text{ is the permanent post of } a_i \\ \Gamma_{i,1}, \Gamma_{i,2}, \ldots, \Gamma_{i,\gamma_i}, \{p'_i\}, & \text{if } a_i \text{ has no permanent post} \end{cases}$$

The *current post* of a_i is p_j and p'_i, respectively. The preference list of post p_j is $a_i, a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$, if p_j is the permanent post of a_i, and $a_1, \ldots, a_n$ if p_j is not assigned to an applicant currently. The preference list of p'_i is a_i (if a_i has no permanent post already).

Let $\text{rank}_{a_i}(p)$ be the rank of p in the extended preference list of applicant a_i, if p belongs to that list. That is, $\text{rank}_{a_i}(p) = l$ iff $p \in \Gamma_{i,l}$.

In TRecruitment we seek for an *applicant-optimal stable matching* of applicants to posts that are in their extended preference lists.

Definition 1. *A matching $M^\star$ is* applicant-optimal *(*a-optimal*, for short) iff $z^\star \leq_{lex} z$ for every matching M, being $z^\star$ and z the sequence of ranks of the posts assigned to $a_1, \ldots, a_n$ in $M^\star$ and M.*

Definition 2. *An applicant-optimal stable matching is a matching that is a-optimal within the set of GS-stable matchings.*

We noticed that, in TRecruitment, we are not searching just for any GS-stable matching but for one that is also applicant-optimal. In our opinion, searching for stable matchings that assign the best posts to the highest ranked applicants was implicit in the law. The problem may be modelled as a constraint satisfaction and optimization problem:

TRecruitment
  minimize_lexicographically $z = (z_1, \ldots, z_n)$
  subject to
    $y_i \in dom(y_i)$, $z_i \in dom(z_i)$, $z_i = rank_{a_i}(y_i)$, for $1 \leq i \leq n$
    $y_i \neq y_{i'}$, for $1 \leq i < i' \leq n$
    if $y_{i'} \in dom(y_i)$ and $z_{i'} < \gamma_{i'} + 1$ then $z_i \leq rank_{a_i}(y_{i'})$, for $1 \leq i < i' \leq n$.

using $2n$ decision variables y_i and z_i, for $i = 1, \ldots, n$, with the domains of y_i and z_i given by $dom(y_i) = \cup_{l=1}^{\gamma_i+1} \Gamma_{i,l}$ and $dom(z_i) = \{1, \ldots, \gamma_i + 1\}$, the value of y_i giving the final post of a_i and z_i its rank in a_i's extended preference list. The constraints state that applicants must get matched to distinct posts and the matching must be GS-stable.

In TRecruitment, we call a matching is *TR-stable* iff it is GS-stable and the permanent posts do not prevent their owners from moving. This notion is formally defined as follows.

Definition 3. *M is TR-stable iff M is GS-stable and there is no GS-stable matching M' and some $i \geq 1$ such that $a_1, \ldots, a_{i-1}$ are matched to exactly the same posts in M and M' and a_i is matched to his permanent post in M but not in M'.*

Applicant-optimal stable matchings are a-optimal TR-stable also.

Example 2. In case B, $\{(a_1, p_1), (a_2, p_2), (a_3, p_3)\}$ is GS-stable but not TR-stable. The matching $\{(a_1, p_3), (a_2, p_2), (a_3, p_1)\}$ is a-optimal but not GS-stable and $\{(a_1, p_2), (a_2, p_1), (a_3, p_3)\}$ is a-optimal stable.

Case A	**Case B**
$a_1 : \{p_1, p_2, p_3, p_4\}, \{p'_1\}$	$a_1 : \{p_3\}, \{p_2\}, \{p_1\}$
$a_2 : \{p_1, p_2, p_3, p_4\}, \{p'_2\}$	$a_2 : \{p_1\}, \{p_2\}$
$a_3 : \{p_1\}, \{p'_3\}$	$a_3 : \{p_1\}, \{p_3\}$
$a_4 : \{p_2\}, \{p'_4\}$	

In case A, $\{(a_1, p_1), (a_2, p_2), (a_3, p'_3), (a_4, p'_4)\}$ is TR-stable but not a-optimal and $\{(a_1, p_3), (a_2, p_4), (a_3, p_1), (a_4, p_2)\}$ is a-optimal stable.

Hence, TRecruitment seeks for a matching where every a_i gets a post among the best he may get in all TR-stable matchings that guarantee the same for $a_1, \ldots, a_{i-1}$, which seems fair.

3 Preferences in Strict Order

When every $\Gamma_{i,l}$ is a singleton for all i and l, the preference lists of applicants are totally ordered. The main idea of the $O(\max(1, t)m)$ algorithm given previously was to go on assigning the best free posts to applicants, following the order $a_1, \ldots, a_n$. If in the end, an applicant who had a permanent position resulted unmatched, then its position and the applicant were removed, and the procedure was repeated from the beginning. Clearly, the Gale-Shapley Algorithm [5, 9] may be used instead to find applicant-optimal stable matchings in $O(m)$.

To apply this algorithm, we consider the extended preference lists for applicants who initially had posts and the true preference lists for the remaining ones. The preference lists of schools are the rank list of applicants, except that each applicant who has a permanent post in some school is top-rank for that school.

For the case of totally ordered preference lists, our assumption that schools do not try to retain teachers essentially corresponds to see as inadmissible every GS-stable matching that is applicant non-optimal. As a consequence, we have the following result.

Proposition 1. *In case the preference lists of applicants are totally ordered,* TRecruitment *admits a unique TR-stable matching. This matching is applicant-optimal and may be found in $O(m)$ by* Gale-Shapley Algorithm.

Proof. It is known that any instance of Stable Marriage with strictly ordered preference lists, in which all women (that is, posts/schools) have exactly the same preference list, admits a unique GS-stable matching [9]. Based on this, we know that if there were no tenured posts, applicant i gets matched to his favourite post among those that remain free after matching $a_1, \ldots, a_{i-1}$. Since in TRecruitment posts prefer their owners, they do not have exactly the same preference lists. Nevertheless, the least preferred post for a_i is his current post. So, for a moment, let us assume that all posts have the same preference list, namely $a_1, \ldots, a_n$, and that $\Gamma_{i,1}, \Gamma_{i,2}, \ldots, \Gamma_{i,\gamma_i}$ is the preference list of a_i, for all i. If we apply that sequential method and find no post left for a_i when we reach a_i's turn, there is no GS-stable matching in which a_i succeeds in moving from his current post. If a_i had a permanent post p_j, the matching is infeasible. We may remove a_i and p_j and restart the procedure from the beginning. This shows uniqueness. That the matching is a-optimal follows from a classical result of Stable Marriage with incomplete lists (no ties) [5, 9]: each instance admits a unique a-optimal GS-stable matching and Gale-Shapley Algorithm (applicant-oriented) finds it in $O(m)$. □

4 Preferences with Ties

As a result of Proposition 1, we clearly see that there is some risk that some applicants result unfairly unmatched if posts are sorted arbitrarily in strict order to break ties (see Example 2, case A). General instances of Stable Marriage, with strictly ordered preference lists on both sides, may have an exponential number of GS-stable matchings [14]. For Stable Marriage with incomplete lists (no ties), no matter which algorithm we apply, exactly the same applicants and exactly the same posts get matched, although the resulting matchings may be distinct. This does not hold for Stable Marriage with ties and incomplete lists [10, 17]. Because of that, we search for applicant-optimal stable matchings to achieve fair matchings that applicants cannot contest. The pragmatics here is whether we accept any TR-stable solution as fair due to computational costs, or just the solutions that are also applicant-optimal, as for TRecruitment. We shall now show that TRecruitment may be found by a strong polynomial algorithm.

4.1 Polynomial Complexity

Our first proof that TRecruitment was polynomially solvable was based on its reduction to a sequence of at most t minimum-weight perfect match-

ing problems in a bipartite graphs, where t is the number of applicants with tenured posts [24]. To reduce the problem we introduced *edge weights* that guarantee GS-stability but are exponential. Indeed, we consider a relaxation of TRecruitment where we no longer require that applicants that had posts initially should be matched in the end to a real post. Thus, we extend the preference list of a_i:

$$a_i : \begin{cases} \Gamma_{i,1}, \Gamma_{i,2}, \ldots, \Gamma_{i,\gamma_i}, \{p_j\}, \{p'_i\} & \text{if } p_j \text{ is the permanent position of } a_i \\ \Gamma_{i,1}, \Gamma_{i,2}, \ldots, \Gamma_{i,\gamma_i}, \{p'_i\}, & \text{if } a_i \text{ has no permanent position} \end{cases}$$

Denoting by Ω_i the rank of the last resort post, in each case, i.e., γ_i+2 and $\gamma_i + 1$, respectively, we then define the weight $c(a_i, \Gamma_{i,l})$ of each positions that belongs to $\Gamma_{i,l}$ by

$$c(a_i, \Gamma_{i,t}) = (l-1)r_i$$

that is $c(a_i, p) = (l-1)r_i$, for all $p \in \Gamma_{i,l}$ with $1 \leq l \leq \Omega_i$. The chosen values $c(a_i, \Gamma_{i,l})$ are terms of an arithmetic progression with ratio r_i and first term 0. For the ratio we choose

$$r_n = 1, \quad r_{i-1} = r_i \Omega_i = \textstyle\prod_{q=i}^{n} \Omega_q \text{ for all } 1 < i \leq n \tag{1}$$

that ensures that minimum weight perfect matchings are TR-stable and applicant-optimal in the extended bipartite graph G', as we state in Proposition 2 below. The matching may be inadmissible, if it violates the constraints imposed by tenure, but, nevertheless, may give important information for achieving an optimal feasible matching.

We introduce also ficticious applicants to be able to restrict search to perfect matchings. So fictitious applicants, we define $c(a'_j, p_j) = 0$, for $1 \leq j \leq k$ and $c(a'_j, p'_i) = 0$, for $1 \leq j \leq k$ and $1 \leq i \leq n$. The weight function c has been chosen in such a way that no group of applicants of a smaller rank may violate TR-stability to globally reduce the sum of weights of positions they get matched to.

Proposition 2. *If no applicant has a tenure (i.e., $G = G'$), the optimal solutions of* MinWeightPerfectMatching(G') *are a-optimal and TR-stable, for the cost function c and graphs G and G' defined above.*

MinWeightPerfectMatching is a classical assignment problem and may be solved in strong polynomial time, for instance by Kuhn-Munkres algorithm (a.k.a, Hungarian method) [3, 16, 22]. For n-vertex bipartite graphs, quite simple implementations of this method run in $O(n^4)$ but this complexity bound can be reduced to $O(n^3)$ for example [3, 22]. This would lead to $O(\max\{1, t\}n(n+k)^3)$ time complexity bound for TRecruitment [24].

4.2 Related Work and Better Complexity Bounds

A better algorithm may be obtained if each weighted matching problem is solved using a scaling algorithm[1]. In fact, under the assumption that the input costs are integers in the range $[-C..C]$, Gabow and Tarjan [4] use cost scaling and blocking flow techniques to obtain an $O(\sqrt{|V|}|E|\log(|V|C))$ time algorithm for the assignment problem, where V and E are the sets of nodes and edges of the graph. In [7], Goldberg and Kennedy study implementations of scaling push-relabel methods, having the same running time bound, and investigate heuristics to improve their performance in practice.

By using these scaling methods for solving each optimal weight matching problem, we may solve TRecruitment in $O(\max\{1,t\}n(n+k)^{2.5})$, assuming that $\max_i(\gamma_i) \leq U$ and $O(U)$ arithmetic may be done in constant time. If $\max_i(\gamma_i)$ is $O(k)$, then the running time complexity would be $O(\max\{1,t\}n(n+k)^{2.5}\log k)$. Indeed, for reducing TRecruitment to a sequence of maximum weight matchings, we have defined a cost function that guarantees the stability of the optimal matching (i.e., that the priorities are not violated). Unfortunately, the introduced edge costs may be of $O(k^n)$.

This kind of reduction to weighted matching was previously independently introduced in [11, 12] by Irving et *al*, for solving *greedy (rank-maximal) matching problems*. It is worth noting that *applicant-optimal stable matchings* for TRecruitment are not greedy matchings in general.

The fact that these reductions to weighted matching involve huge edge costs (weights), led Mehlhorn and Michail [19] to exploit further the scaling method introduced in [12] to handle huge weights efficiently (i.e., in constant time). This is achieved by exploiting the properties of the potential function and reduced costs involved on these primal-dual algorithms. It has advantages both in terms of space and time complexity. It is possible to reduce TRecruitment complexity based on the same techniques. However, we have developed a simpler method that does not require the use of a cost function. We describe it in detail in the section 5.

Some ideas for this new algorithm came from the analysis of the Gale-Shapley Algorithm and the techniques used in the scaling algorithms [1, 12]. We solve TRecruitment as a sequence of maximal cardinality bipartite matching problems on reduced subgraphs, combined with an effective propagation of the stability constraints. At each stage, we try to

[1] I would like to thank Dimitrios Michail for this suggestion.

augment a partial matching for provisionally (or even definitely) matching one more applicant, sequentially considering the highest ranked posts still eligible (i.e., that are not definitely discarded by the propagation of the stability constraints). Applicants are taken by decreasing ranks.

5 An Efficient Algorithm for Solving TRecruitment

We reduce TRecruitment to a sequence of maximum cardinality matchings. In each iteration, only the applicants up to position i are handled, for increasing values of i from 1 to n.

The idea is as follows. We start from applicant 1 and match him to a post in his first choice. We add all edges (a_1, p), for all $p \in \Gamma_{1,1}$ to the bipartite graph. Then, we consider applicant 2 and try to augment the *witness partial matching* restricting to the posts in his first choice also (i.e., to p's in $\Gamma_{2,1}$), analysing them one by one. For the analysis of post p, we may apply a classical algorithm for augmenting a matching [22]. If there is no augmenting (alternating) path, post p records that its acceptance level is less than i (in this case $i = 2$). If there is an augmenting path, we add edges from applicant a_i to all its first rank choices, extending the graph. The algorithm proceeds with the next applicant.

If, when we are trying to assign a post to a given applicant a_i, we find that there remain no choices for a_i, then if a_i had no permanent post, a_i becomes definitely unmatched.

If a_i had a permanent post, say p, that is assigned to some a_j for $j < i$, then a_j becomes unmatched to match a_i definitely to p. Then, we start a *repairing phase* to assign a new post to a_j. Let z_j be the level of preference at which a_j was previously matched (that is equal to the preference of a_j for post p).

We first try to repair the matching without increasing z_j. This may be done only if we find an *augmenting alternating path* that starts at a_j and ends at some $a_{j'}$ with $j' > j$. Then, $a_{j'}$ becomes unassigned, and we go on repairing, now $a_{j'}$. Without increasing the complexity of the algorithm, we may even select $a_{j'}$ as the lowest ranked (provisionally) assigned applicant that is accessible from a_j in the Hungarian tree.

If we cannot find an augmenting path using posts in Γ_{j,z_j}, the value of z_j is definitely increased (by one unit). All edges incident to a_j are removed and we start considering posts in the new Γ_{j,z_j}, to augment the matching. Again, these posts are considered one by one so that we may record no-goods for subsequent steps. This is important for guaranteeing the stability of the matching. In this case, the matching may

be augmented if the augmenting path ends at a post that is free or at a provisionally matched applicant that has smaller rank.

If we cannot find an augmenting path using posts in Γ_{j,z_j}, we go on incrementing z_j, and the algorithm proceeds in a similar way. While we go on searching for a post to a_j, the labels on the graph are not removed, as in the classical algorithm for augmenting a matching [22].

5.1 Extensions to Capacitated Assignments

It is important for the real-life application that we consider the assignment of applicants to schools rather than to posts. Each school may have a number of vacancies, all corresponding to identical posts. Due to demographic reductions, some schools in Portugal have *negative vacancies* nowadays. The law says that whenever a teacher that detains such a vacancy applies for another permanent position and gets it, his vacancy cannot be assigned to another candidate. In this way, a school that has negative vacancies may admit new teachers only if the tenured applicants that move from it outnumbers the applicants that were assigned a post in it.

The algorithm given previously may be easily adapted to solve this capacitated TRecruitment. As for the previous model, we initially define the number of vacancies of each school as the number of the published vacancies plus the number of applicants that are trying to move from it. An *augmenting path* terminates at a school that has a positive number of vacancies or at a provisionally assigned applicant of lower rank. When an applicant a_i with tenure has to be definitely assigned to his initial position and there are no longer vacancies at his school, the lowest ranked applicant provisionally assigned to that school becomes unmatched, if there is any. Otherwise, a_i is definitely assigned to his school, which necessarily ends up with a negative number of vacancies.

References

1. A. Abdulkadiroğlu, T. Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, **88** (1999) 233–260.
2. D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, SIAM (2005) 424–432.
3. W. Cook, W. Cunningham, W. Pulleyblank, A. Schrijver. *Combinatorial optimization*. John Wiley & Sons, 1998.
4. H. N. Gabow, R. E. Tarjan. Almost-optimal speed-ups of algorithms for matching and related problems. In *Proceedings 20th Annual ACM Symposium on Theory of Computing* (1988) 514–527.

5. D. Gale, L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly* **69** (1962) 9–15.
6. P. Gardenfors. Match Making: assignment based on bilateral preferences. *Behavioural Sciences*, **20** (1975), 166–173.
7. A. V. Goldberg, R. Kennedy. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming* **71** (1995) 153–177.
8. D. Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM J. Computing* **16(1)** (1987) 111–128.
9. D. Gusfield, R. W. Irving. *The stable marriage problem – structure and algorithms*. MIT Press, 1989.
10. R. Irving. Stable marriage and indifference. *Discrete Applied Mathematics* **48** (1994) 261–272.
11. R. W. Irving. Greedy matchings. University of Glasgow, Computing Science Department Research Report, TR-2003-136, April 2003
12. R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Rank-maximal matchings. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, SIAM (2004) 68–75.
13. K. Iwama, D. Manlove, S. Miyazaki, Y. Morita. Stable marriage with incomplete lists and ties. In J. Wiedermann, P. van Emde Boas, M. Nielsen (Eds) *Automata, Languages and Programming, 26th International Colloquium, ICALP'99*, Lecture Notes in Computer Science **1644**, Springer-Verlag (1999) 443–452.
14. R. Irving, P. Leather. The complexity of counting stable marriages. *SIAM J. Computing* **15** (1986) 655–667.
15. D. E. Knuth. *Stable marriage and its relation to other combinatorial problems*. CRM Proceedings and Lecture Notes **10**, AMS, 1997.
16. H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quaterly* **2** (1955) 83–97 (republished in *Naval Research Logistics* **52(1)** (2005) 7–21).
17. D. F. Manlove, R. Irving, K. Iwama, S. Miyazaki, Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science* **276** (2002) 261–279.
18. D. F. Manlove and C. T. S. Sng. Popular matchings in the capacitated house allocation problem. In Y. Azar, T. Erlebach (Eds). *Algorithms - ESA 2006*, Lecture Notes in Computer Science **4168**, Springer-Verlag (2006) 492–503.
19. K. Mehlhorn, D. Michail. Network problems with non-polynomial weights and applications. Max-Planck Institute (2006), Germany (unpublished draft).
20. J. Mestre. Weighted popular matchings. In M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.) *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Proceedings, Part I*, Lecture Notes in Computer Science **4051**, (2006) 715–726
21. J. Munkres. Algorithms for the assignment and transportation problems. *SIAM Journal on Applied Mathematics* **5** (1957) 32–38.
22. L. A. Wolsey. *Integer programming*. Wiley-Interscience, 1998.
23. Y. Yuan. Residence exchange wanted: a stable residence exchange problem. *European Journal of Operational Research* **90** (1996) 536–546.
24. A. P. Tomás. Emparelhamentos, casamentos estáveis e algoritmos de colocação de professores. Technical Report DCC-2005-02, DCC - FC & LIACC, University of Porto, 2005 (in Portuguese). `www.dcc.fc.up.pt/Pubs/TR05/dcc-2005-02.pdf`