Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 155 (2007) 2118–2129

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

The stable fixtures problem—A many-to-many extension of stable roommates

Robert W. Irving*, Sandy Scott[1]

Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, Scotland

Received 25 March 2002; received in revised form 18 July 2006; accepted 16 May 2007
Available online 24 May 2007

Abstract

We study a many-to-many generalisation of the well-known stable roommates problem in which each participant seeks to be matched with a number of others. We present a linear-time algorithm that determines whether a stable matching exists, and if so, returns one such matching.

Keywords: Stable matching problem; Many-to-many non-bipartite matching; Linear-time algorithm

1. Introduction

The *stable fixtures* (SF) *problem* is a generalisation of the stable roommates (SR) problem (a detailed treatment of which can be found in [4, Chapter 4]) in which each participant has a fixed *capacity*, and is to be assigned a number of matches less than or equal to that capacity subject to the normal stability criterion. The name derives from a possible application in which a set of players (or teams) take part in a competition in which the *fixtures*, in the form of a set of matches between players, are to be specified in advance. Each player has a specified target number of matches, and may play against each of the others at most once. Each ranks a subset of the others—his acceptable opponents—in order of preference. A set of fixtures is stable if there are no two mutually acceptable players, who do not form a match, but each of whom either prefers the other to one of his prescribed matches or has fewer matches than his capacity.

More formally, an instance of the SF problem consists of

- $\mathscr{X} = \{x_1, \ldots, x_n\}$, the set of players;
- for each i $(1 \leqslant i \leqslant n)$ a positive integer c_i, which we refer to as the *capacity* of x_i; this is the maximum number of matches that x_i can have;
- a *preference structure*, denoted by P; this comprises a *preference list* P_i for each x_i $(1 \leqslant i \leqslant n)$, which is a strictly ordered subset of $\mathscr{X} \backslash \{x_i\}$.

* Corresponding author.
E-mail addresses: rwi@dcs.gla.ac.uk (R.W. Irving), sas@dcs.gla.ac.uk (S. Scott).
[1] Supported by University of Glasgow 2001 Postgraduate Research Scholarship.

0166-218X/$ - see front matter
doi:10.1016/j.dam.2007.05.015

If x_j appears in P_i we say that x_j is *acceptable* to x_i. If x_j precedes x_k in P_i we say that x_i *prefers* x_j to x_k. A pair $\{x_i, x_j\}$ is an *acceptable pair* if x_i is acceptable to x_j, and x_j is acceptable to x_i. Only acceptable pairs play a role in the SF problem, so we may assume, without loss of generality, that if x_j is acceptable to x_i but x_i is not acceptable to x_j, then x_j is simply deleted from P_i, and hence that the preference lists are *consistent*, i.e., x_j is in P_i if and only if x_i is in P_j. We denote by $|P_i|$ the number of entries in the list P_i.

A *matching* $\mathscr{M}$ is a set of acceptable pairs $\{x_i, x_j\}$ such that, for all i $(1 \leqslant i \leqslant n)$,

$$|\{x_j : \{x_i, x_j\} \in \mathscr{M}\}| \leqslant c_i.$$

The *size* of $\mathscr{M}$ is the number of pairs in $\mathscr{M}$. The members of the set $\mathscr{M}(x_i) = \{x_j : \{x_i, x_j\} \in \mathscr{M}\}$ are referred to as the *matches* of x_i in $\mathscr{M}$.

An acceptable pair $\{x_i, x_j\} \notin \mathscr{M}$ is a *blocking pair* for matching $\mathscr{M}$, or *blocks* $\mathscr{M}$, if

- either x_i has fewer than c_i matches or prefers x_j to at least one of his matches in $\mathscr{M}$; and
- either x_j has fewer than c_j matches or prefers x_i to at least one of his matches in $\mathscr{M}$.

A matching for which there is no blocking pair is said to be *stable*, and is otherwise *unstable*.

The case in which $c_i = 1$ for all i is the well-known SR problem. SR was introduced by Gale and Shapley [2] in their classical paper on the *stable marriage* (SM) problem. They showed that, in contrast to the case of SM, an instance of SR may or may not admit a stable matching. Subsequently, the possibility of a polynomial-time algorithm to solve SR was presented as an open problem by Knuth [8]. This question was resolved by Irving [5], who gave a polynomial-time algorithm to determine whether a stable matching exists, and if so to find one such matching. Further algorithmic results for SR were established by Gusfield [3], and these various results were collated and extended in [4].

In fact, in the classical version of SR, n is even and all preference lists are *complete*, i.e., $x_j \in P_i$ for all $i, j, i \neq j$, so that if a stable matching does exist then every player is matched to exactly one other. So in this context, a matching is a *perfect matching*. If n is odd and/or preference lists may be incomplete, then a stable matching, if one exists, may or may not be perfect [4]. However, if a stable matching does exist, then every stable matching involves the same set of players, and as a consequence, all stable matchings have the same size [4].

By contrast, in the SF problem, it is easy to construct an example to show that, even if all preference lists are complete, a stable matching may have some players with fewer matches than their capacity. For example, consider an instance involving four players, each of capacity 2, in which one particular player is ranked last by each of the others. It is easy to see that the sole stable matching here has size 3, and the unpopular player has no matches. So, at least in this respect, SF behaves somewhat differently from SR.

The SR problem is a generalisation of the SM problem, in the sense that, for any instance of SM there is an instance of SR involving the same players for which exactly the same matchings are stable [4]. The many-to-one generalisation of SM, the so-called Hospital-Residents (HR) problem, together with its important practical applications, has been extensively studied—see, for example, [4,11]. Recently Baïou and Balinski [1] have shown that the many-to-many extension of SM can be solved in $O(n^2)$ time, where n is the size of the larger set. The SF problem is simultaneously a generalisation of all of these others, and therefore has considerable theoretical interest as ostensibly the most general of all stable matching problems.

Here we present an extension of Irving's SR algorithm which, for a given instance of the SF problem, determines if a stable matching exists, and if so finds one such matching, all in $O(m)$ time, where m is the sum of the lengths of the preference lists. As in the SR case, the SF algorithm is split into two phases, discussed in Sections 2 and 3, respectively. In Section 4 we consider the implementation and complexity of the algorithm, and in Section 5 we present our conclusion and some open problems.

2. Phase 1 of the algorithm

The first phase of the algorithm closely resembles the first phase of the SR algorithm [5], in that, conceptually, it involves a sequence of *bids* from one player for another (usually referred to as 'proposals' in the SM context). These bids enable the construction of a set S of potential matched pairs, and also allow the identification of some pairs that cannot belong to any stable matching, and which are therefore deleted from the preference structure. By the *deletion* of a pair $\{x_i, x_j\}$, we mean the removal of x_i from P_j and of x_j from P_i. Henceforth we denote by P the *current* preference

```
S = ∅;
while a_i < min(c_i, |P_i|)
{
    x_j = first player in P_i who is not in A_i;
    /*  x_i bids for x_j and x_j becomes a target of x_i  */
    S = S ∪ {(x_i, x_j)};
    if b_j ≥ c_j
    {
        x_k = c_j th ranked bidder for x_j;
        for each successor x_l of x_k in P_j
        {
            if (x_l, x_j) ∈ S
                /*  x_j rejects x_l's bid and x_j is no longer a target of x_l  */
                S = S \ {(x_l, x_j)};
            delete the pair {x_l, x_j} from P;
        }
    }
}
```

Fig. 1. Algorithm SF-Phase1.

structure, noting that this changes during the algorithm as a result of deletions. We say that $\{x_i, x_j\}$ is *in P* if x_i is in P_j (or, equivalently, x_j is in P_i since preference structures are always consistent.)

The set S consists of *ordered* pairs (x_i, x_j), and is initially empty. For each player x_i we define two sets A_i and B_i as follows:

$$A_i = \{x_j : (x_i, x_j) \in S\},$$

$$B_i = \{x_j : (x_j, x_i) \in S\}.$$

In terms of bids, A_i represents the set of players for whom x_i has made a bid that has not (yet) been rejected, while B_i represents the set of players who have bid for him and whom he has not rejected. If $x_j \in A_i$ (or equivalently, $x_i \in B_j$) at some point during the algorithm's execution, then we say that x_j is, at that moment, a *target* for x_i and x_i is a *bidder* for x_j. We denote $|A_i|$ and $|B_i|$ by a_i and b_i, respectively, so a_i is the current number of targets of x_i, and b_i the current number of bidders for x_i.

Each successive bid is made by some player x_i who has fewer targets than his capacity; x_i will bid for the first player x_j in P_i who is not already among his targets, and as a result (x_i, x_j) is added to S and x_j becomes a target of x_i. If it is still the case that $b_j < c_j$, i.e., x_j has fewer bidders than his capacity, then no further action results. However, if (i) $b_j = c_j$ or (ii) $b_j = c_j + 1$, the only other possibilities, then in the ranking of x_j's bidders induced by P_j, the bidder ranked in position c_j, say x_k, is identified, and all the pairs $\{x_l, x_j\}$, such that x_j prefers x_k to x_l, are deleted. These deletions might include one pair in S, namely in case (ii) above, and if so this pair is deleted from S—it represents the rejection of x_l's bid by x_j for the x_l in question—and as a result, x_l is no longer a bidder for x_j and x_j is no longer a target for x_l. An immediate invariant is that, for each i, the set A_i consists of the first a_i players in P_i. This phase of the algorithm terminates when, for all i, $a_i = \min(c_i, |P_i|)$.

The algorithm, which we refer to as Algorithm SF-Phase1, is displayed in Fig. 1.

Algorithm SF-Phase1 is non-deterministic but, as with other proposal-based algorithms for solving variants of the SM problem, this non-determinism has no effect on the outcome. We shall call the preference structure resulting from the execution of phase 1 the *phase* 1 *preference structure*, denoted by P^1. For a given player x_i, we denote x_i's list in P^1 by P_i^1. We also denote by S^1, A_i^1 and B_i^1 the sets S, A_i and B_i, and a_i^1, b_i^1 the values of a_i, b_i, respectively, for each i, on termination of SF-Phase1.

We are now in a position to draw some conclusions from the outcome of Algorithm SF-Phase1. To this end, we define the terms *stable pair* to be a pair $\{x_i, x_j\}$ that belongs to some stable matching, and *stable match* of a player x_i to be a player x_j such that $\{x_i, x_j\}$ is a stable pair.

Lemma 2.1. *If $\{x_i, x_j\}$ is not in P^1 then $\{x_i, x_j\}$ is not a stable pair.*

Proof. Suppose, for a contradiction, that $\{x_i, x_j\}$ is a stable pair that is not in P^1. Let $\mathscr{M}$ be a stable matching containing the pair $\{x_i, x_j\}$, and suppose that $\{x_i, x_j\}$ was the first stable pair to be deleted during a particular execution E of SF-Phase1. Suppose further, without loss of generality, that the deletion of $\{x_i, x_j\}$ took place when some player x_k bid for x_i. Then x_i must at that point have had $c_i - 1$ existing bidders whom he preferred to x_j, say $x_{i_1}, \ldots, x_{i_{c_i-1}}$, and he must also prefer x_k to x_j. Let U be the set $\{x_{i_1}, \ldots, x_{i_{c_i-1}}, x_k\}$. Not all of the players in U can be matches of x_i in $\mathscr{M}$, for $x_j \notin U$ is one such match. So choose $x_l \in U$ such that $\{x_i, x_l\} \notin \mathscr{M}$. Suppose x_l prefers all of his matches in $\mathscr{M}$ to x_i. Then for x_i to have become a target of x_l during E, some stable pair (involving x_l) must already have been deleted, contradicting the assumption that $\{x_i, x_j\}$ was the first such deletion. It follows that x_l prefers x_i to at least one of his matches in $\mathscr{M}$, and we know that x_i prefers x_l to x_j, a contradiction of the stability of $\mathscr{M}$. □

The following corollary is immediate.

Corollary 2.1. *Any player x_i for whom $|P_i^1| = 0$ has no stable matches.*

Some further lemmas record various properties of stable matchings, should any exist.

Lemma 2.2. *If $(x_i, x_j) \in S^1$ and $(x_j, x_i) \in S^1$ then $\{x_i, x_j\}$ is in every stable matching.*

Proof. Suppose that $(x_i, x_j) \in S^1$ and $(x_j, x_i) \in S^1$, and that $\mathscr{M}$ is a stable matching such that $\{x_i, x_j\} \notin \mathscr{M}$. The fact that $(x_i, x_j) \in S^1$ implies that x_j is among the first c_i entries in P_i^1, and likewise $(x_j, x_i) \in S^1$ implies that x_i is among the first c_j entries in P_j^1. By Lemma 2.1, x_j cannot have a match in $\mathscr{M}$ who is not in P_i^1, so either he has fewer than c_i matches, or has a match to whom he prefers x_j. A similar observation applies to x_j, and it follows that $\{x_i, x_j\}$ is a blocking pair for $\mathscr{M}$—a contradiction. □

Lemma 2.3. *For all i $(1 \leqslant i \leqslant n)$, $a_i^1 = \min(c_i, |P_i^1|) = b_i^1$.*

Proof. By definition, $a_i = \min(c_i, |P_i^1|)$ for all i is the termination condition of SF-Phase1. It is clear that $\sum a_i^1 = \sum b_i^1$, since each pair in S^1 contributes exactly one to each of these sums. If, for some k, $a_k^1 \neq b_k^1$ then either $a_k^1 < b_k^1$ or $a_k^1 > b_k^1$. In the former case, we let index i be k. In the latter case, because of the equal sums, there must be some $t \neq k$ such that $a_t^1 < b_t^1$, and in that case we let index i be t. So we can assume, without loss of generality, that $a_i^1 < b_i^1$. It follows from the algorithm that $b_i^1 \leqslant c_i$, since whenever b_i becomes greater than c_i, x_i rejects one of his bidders. Hence $a_i^1 < c_i$, so that $a_i^1 = |P_i^1|$. For each $x_j \in B_i^1$, by definition we have $(x_j, x_i) \in S^1$, and so $x_i \in P_j^1$, and $x_j \in P_i^1$ by consistency. It follows that $b_i^1 = |B_i^1| \leqslant |P_i^1| = a_i^1$, a contradiction. □

Lemma 2.4. *Let x_i be a player for whom $|P_i^1| \leqslant c_i$. Then in every stable matching, the matches of x_i are precisely the players x_j in P_i^1.*

Proof. By Lemma 2.1, no player who is not in P_i^1 can be a stable match of x_i. Also, for every x_j in P_i^1, we must have $(x_i, x_j) \in S^1$, otherwise $a_i^1 \neq \min(c_i, |P_i^1|)$, and the termination condition for SF-Phase1 is not satisfied. Since $a_i^1 = b_i^1$, by Lemma 2.3, and since the only candidates for the set B_i^1 are the a_i entries on P_i^1, it follows that all of these entries are in B_i^1, and therefore $(x_j, x_i) \in S^1$ for all x_j in P_i^1. The result follows from Lemma 2.2. □

Lemma 2.5. *Any player x_i for whom $|P_i^1| \geqslant c_i$ must have exactly c_i matches in any stable matching.*

Proof. It must be the case that $a_i^1 = c_i$, otherwise the termination condition $a_i^1 = \min(c_i, |P_i^1|)$ would not be satisfied, and therefore, by Lemma 2.3, $b_i^1 = c_i$ also.

Suppose that $B_i^1 = \{x_{i_1}, \ldots, x_{i_{c_i}}\}$. Then for each j $(1 \leqslant j \leqslant c_i)$ x_i appears in the first c_{i_j} positions in $P_{i_j}^1$. If, in a matching $\mathscr{M}$, x_i has fewer than c_i matches then, in particular, one of $x_{i_1}, \ldots, x_{i_{c_i}}$, say x_{i_k}, is not a match. So $\{x_i, x_{i_k}\}$ blocks $\mathscr{M}$, a contradiction. □

Initial preference structure		Phase 1 preference structure P^1	
x_1 :(2)	$x_3\ x_2\ x_4\ x_5\ x_7\ x_8\ x_{10}$	x_1 :(2)	$x_3\ x_2\ x_4\ x_5$
x_2 :(2)	$x_1\ x_4\ x_3\ x_5\ x_8\ x_9$	x_2 :(2)	$x_1\ x_4\ x_3\ x_5$
x_3 :(2)	$x_7\ x_8\ x_9\ x_1\ x_2\ x_4\ x_5\ x_{10}$	x_3 :(2)	$x_8\ x_9\ x_1\ x_2\ x_4$
x_4 :(2)	$x_5\ x_3\ x_9\ x_1\ x_8\ x_2$	x_4 :(2)	$x_5\ x_3\ x_9\ x_1\ x_8\ x_2$
x_5 :(2)	$x_2\ x_3\ x_7\ x_1\ x_4\ x_9\ x_6\ x_{10}$	x_5 :(2)	$x_2\ x_1\ x_4\ x_9$
x_6 :(2)	$x_7\ x_9\ x_8\ x_5$	x_6 :(2)	x_7
x_7 :(1)	$x_6\ x_1\ x_8\ x_3\ x_5\ x_{10}$	x_7 :(1)	x_6
x_8 :(1)	$x_1\ x_4\ x_7\ x_9\ x_2\ x_3\ x_6$	x_8 :(1)	$x_4\ x_9\ x_3$
x_9 :(1)	$x_2\ x_5\ x_8\ x_4\ x_3\ x_6$	x_9 :(1)	$x_5\ x_8\ x_4\ x_3$
x_{10}:(1)	$x_1\ x_5\ x_7\ x_3$	x_{10}:(1)	

$S^1 = \{(x_1,x_3), (x_1,x_2), (x_2,x_1), (x_2,x_4), (x_3,x_8), (x_3,x_9), (x_4,x_5),$
$(x_4,x_3), (x_5,x_2), (x_5,x_1), (x_6,x_7), (x_7,x_6), (x_8,x_4), (x_9,x_5)\}$

Fig. 2. The outcome of `SF-Phase1` for an example instance.

We denote $\min(c_i, |P_i^1|)$ by d_i, and refer to this as the *degree* of player i. Note that the degree of a player is a property of the problem instance; it does not change its value in the course of the algorithm.

Theorem 2.1. (i) *The number of matches for a given player x_i is the same in all stable matchings, namely d_i.*
(ii) *All stable matchings for a given instance of the SF problem have the same size.*

Proof. (i) This follows at once from Corollary 2.1 and Lemmas 2.4 and 2.5.
(ii) This is an immediate consequence of part (i). □

Corollary 2.2. *For a given instance of the SF problem, if $\sum d_i$ is odd, then there is no stable matching.*

Proof. By Theorem 2.1 (i), this sum is double the size of a stable matching, since it counts every matched pair exactly twice. □

Example. The original and phase 1 preference structures for an example instance are displayed in Fig. 2, together with the set S^1. Player x_i's preference list is represented in the form $x_i : (c_i)\ P_i$.

It may be verified that $\sum d_i = 14$, so that Corollary 2.2 does not apply in this case. We can conclude that, if a stable matching exists, then it must have size 7. We can make the following additional observations from P^1:

- player x_{10} has an empty list and must therefore be unmatched in every stable matching;
- players x_6 and x_7 have only each other on their lists and so are matched only with each other in every stable matching;
- player x_6, who has a capacity of 2, cannot attain that capacity in any stable matching;
- by Lemma 2.2, players x_1 and x_2 must be matched in every stable matching.

Of course, we do not yet know whether a stable matching exists for this instance—we shall see later that at least one such matching does exist.

For what follows, we need one further result concerning the outcome of `SF-Phase1`.

Lemma 2.6. *If $d_i < c_i$, $d_j < c_j$ and $\{x_i, x_j\}$ is not in P^1, then $\{x_i, x_j\}$ cannot be an acceptable pair.*

Proof. Let us assume that $d_i < c_i$, $d_j < c_j$, and (x_i, x_j) is not in P^1. Then if $\{x_i, x_j\}$ is an acceptable pair, it must have been deleted during the execution of `SF-Phase1`, say when some player x_k became a bidder for x_i. At that point, x_i must have had c_i bidders. Subsequently, x_i can lose a bidder only on gaining another one, so he can never again have fewer than c_i bidders. Hence $|S_i^1| \geqslant c_i$, so that $d_i = \min(c_i, |S_i^1|) = c_i$, a contradiction. □

3. Phase 2 of the algorithm

Our starting point for phase 2 of the algorithm is the set S^1 and preference structure P^1 constructed by phase 1. The essence of phase 2 is the further development of the set S of potential matched pairs, and the further reduction of the preference structure P. Ultimately, we may find that one of the preference lists, say P_i, has its length reduced below the degree d_i of x_i, which is a signal that no stable matching exists. Otherwise, the set S eventually becomes *symmetric*, i.e., $(x_i, x_j) \in S \Leftrightarrow (x_j, x_i) \in S$, in which case the pairs in S constitute a stable matching.

We use the symbol P to represent the (changing) preference structure, starting with $P = P^1$. For phase 2, the set $S = S(P)$ is defined in terms of P as follows:

$$S(P) = \{(x_i, x_j) : x_j \text{ is in the first } d_i \text{ positions in } P_i\}.$$

We call a preference list P_i in P *short* if $|P_i| < d_i$, and *long* if $|P_i| > d_i$. (Note that in case $|P_i^1| = d_i$ it follows that $|P_i^1| \leqslant c_i$, and this is covered by Lemma 2.4.)

The sets A_i and B_i, with sizes a_i and b_i, respectively, are defined as before, namely

$$A_i = \{x_j : (x_i, x_j) \in S(P)\},$$

$$B_i = \{x_j : (x_j, x_i) \in S(P)\}.$$

Note that A_i and B_i depend on S, which in turn depends on P, but we suppress this dependence in the notation for brevity. We continue to refer to A_i as the set of *targets* for x_i, and B_i as the set of *bidders* for x_i. The set A_i consists of the first d_i players in the list P_i.

In addition, we denote by $x_{l(i)}$ the last player in P_i, and by $x_{f(i)}$ the first player, if any, in P_i who is not in A_i, in other words, the player in position $d_i + 1$ in P_i. The player $x_{f(i)}$ is defined only for players x_i with a long preference list. For obvious reasons, we refer to $x_{l(i)}$ as the *worst bidder* for x_i, and $x_{f(i)}$ as the *next target* for x_i.

Throughout phase 2, the preference structure P and its associated set $S(P)$ have certain crucial properties, which are encapsulated in the following definition.

A preference structure P is called *stable* if

SPS1: for all i $(1 \leqslant i \leqslant n)$, $a_i = b_i (= d_i)$;

SPS2: for all i $(1 \leqslant i \leqslant n)$ such that P_i is non-empty, $x_{l(i)} \in B_i$;

SPS3: an acceptable pair $\{x_i, x_j\}$ is not in P if and only if x_i prefers $x_{l(i)}$ to x_j or x_j prefers $x_{l(j)}$ to x_i.

The following lemma shows that phase 2 of the algorithm begins with a stable preference structure.

Lemma 3.1. *The phase* 1 *preference structure* P^1, *with its associated set* S^1, *is a stable preference structure.*

Proof. Property SPS1 is immediate from Lemma 2.3. If $d_i = |P_i^1|$, then property SPS2 follows from the fact that all of the players in P_i^1 are in B_i, and otherwise from the fact that all successors in P_i^1 of the least preferred member of B_i^1 are explicitly deleted during `SF-Phase1`. Finally, Property SPS3 is a consequence of the fact that a pair $\{x_i, x_j\}$ is deleted during `SF-Phase1` only if x_i prefers his current c_ith choice bidder to x_j or x_j prefers his current c_jth choice bidder to x_i, and in either case, the bidder in question becomes the last entry in the preference list. □

The next lemma characterises the preference structure on termination of phase 2 in the case where a stable matching exists.

Lemma 3.2. *Let P be a stable preference structure in which, for all i* $(1 \leqslant i \leqslant n)$, $|P_i| = d_i$. *Then the set* $S = S(P)$ *is symmetric*, i.e., $(x_i, x_j) \in S \Leftrightarrow (x_j, x_i) \in S$; *furthermore, the set of unordered pairs represented by S forms a stable matching.*

Proof. Suppose that $(x_i, x_j) \in S$. This implies that $x_j \in P_i$, and by the consistency of P, that $x_i \in P_j$. The fact that $|P_j| = d_j$, together with property SPS1, implies that all of the players in P_j are in A_j, and so in particular $p_i \in A_j$, hence $(p_j, p_i) \in S$.

Define the matching $\mathscr{M} = \{\{x_k, x_l\} : (x_k, x_l) \in S\}$. Let $\{x_i, x_j\}$ be an acceptable pair that is not in P. Then, by property SPS3, either x_i prefers $x_{l(i)}$ to x_j or x_j prefers $x_{l(j)}$ to x_i. It follows that x_i prefers all of his matches in $\mathscr{M}$ to x_j, or x_j prefers all of his matches in $\mathscr{M}$ to x_i. So the only way that $\{x_i, x_j\}$ can block $\mathscr{M}$ is if each has fewer matches than his capacity. Since, by Theorem 2.1(i), x_i and x_j have d_i and d_j matches, respectively, in $\mathscr{M}$, this implies that $d_i < c_i$ and $d_j < c_j$. Hence, by definition, $d_i = |P_i^1|$ and $d_j = |P_1^j|$, so the acceptable pair $\{x_i, y_j\}$ must have been deleted by Algorithm `SF-Phase1`, and so is not in P^1, in contradiction of Lemma 2.6. □

The key to the further reduction of the preference structure is the concept of a rotation, similar to that introduced by Irving [5] for the SR algorithm. Relative to a preference structure P, a *rotation* is a sequence of ordered pairs

$$\rho = ((x_{i_0}, x_{j_0}), (x_{i_1}, x_{j_1}), \ldots, (x_{i_{r-1}}, x_{j_{r-1}})),$$

where, for each k $(0 \leqslant k \leqslant r-1)$,

$$x_{i_k} = x_{l(j_k)} \quad \text{and} \quad x_{j_{k+1}} = x_{f(i_k)},$$

with $k+1$ evaluated modulo r. So x_{i_k} is x_{j_k}'s worst bidder, and $x_{j_{k+1}}$ is x_{i_k}'s next target. For each k, we say that x_{i_k}, x_{j_k} and the pair (x_{i_k}, x_{j_k}) are *in* the rotation ρ.

Example. In the example illustrated in Fig. 2, it may be verified that, for the preference structure P^1, $\rho = ((x_4, x_3), (x_3, x_9), (x_5, x_1), (x_2, x_4))$ is a rotation, in fact, as it turns out, the only rotation.

The intuition underlying the concept of a rotation is as follows. If $\rho = ((x_{i_0}, x_{j_0}), \ldots, (x_{i_{r-1}}, x_{j_{r-1}}))$ is a rotation, then suppose that, for some k $(0 \leqslant k \leqslant r-1)$, x_{i_k}'s bid for x_{j_k} were to be rejected. Then x_{i_k} would bid for $x_{j_{k+1}}$, causing $x_{i_{k+1}}$'s bid for $x_{j_{k+1}}$ to be rejected. Hence rejections and new bids would follow all the way round the cycle. For each k, we refer to x_{i_k} as the *new bidder* for $x_{j_{k+1}}$.

Lemma 3.3. *Let P be a stable preference structure in which some player x_i has a long preference list. Then there is at least one rotation in P.*

Proof. We denote by $V(P)$ the set of players with long preference lists, i.e., the players x_i for whom $|P_i| > d_i$. For a player $x_i \in V(P)$, let $x_j = x_{l(i)}$, so that $(x_i, x_j) \notin S$. Player x_j is also in $V(P)$, for $(x_i, x_j) \notin S$ implies that there is some k such that $(x_j, x_k) \notin S$, otherwise it could not be the case that $a_j = b_j$, and this is all that is required to ensure that $x_j \in V(P)$. Hence such an x_k is defined, and by the same argument, $x_k \in V(P)$. Since such an x_k exists, $x_{f(j)} = x_{f(l(i))}$ exists, and we will assume for the rest of the proof that$x_k = x_{f(j)}$.

Let $\mathscr{D}(P)$ be a directed graph with a node for each player x_i in $V(P)$. For each node x_i in $\mathscr{D}(P)$, let there be an outgoing edge to the node $x_{f(l(i))}$, which, as we have seen, is well defined and in $V(P)$. Because every node in $\mathscr{D}(P)$ has out-degree 1, $\mathscr{D}(P)$ must contain at least one cycle. Suppose that the nodes in such a cycle are $x_{j_0}, \ldots, x_{j_{r-1}}$ in that order. Then, since $x_{j_{k+1}} = x_{f(l(j_k))}$ for $0 \leqslant i \leqslant r-1$, $k+1$ taken modulo r, it follows that $\rho = ((x_{i_0}, x_{j_0}), \ldots, (x_{i_{r-1}}, x_{j_{r-1}}))$, is a rotation in P, where $x_{i_k} = x_{l(j_k)}$ for all k. □

The proof of Lemma 3.3 provides a means of finding a rotation in a stable preference structure P: starting from any player x_i who has a long preference list, traverse the unique path in $\mathscr{D}(P)$ from the node x_i until some node is visited twice. If ρ is the rotation generated by this traversal, we say that x_i *leads to ρ in P*. If x_i leads to ρ in P, but the node x_i is not itself in the cycle, then the path in $\mathscr{D}(P)$ from the node x_i to the first node in the cycle is called a *tail* of the rotation. For $\rho = (x_{i_0}, x_{j_0}), \ldots (x_{i_{r-1}}, x_{j_{r-1}})$, we refer to $\{x_{i_0}, \ldots, x_{i_{r-1}}\}$ as the *bidder set* and $\{x_{j_0}, \ldots, x_{j_{r-1}}\}$ as the *target set* of ρ. It is easily verified that $x_{i_k} \neq x_{i_l}$ and $x_{j_k} \neq x_{j_l}$ whenever $k \neq l$ (but the bidder set and target set need not be disjoint, as in our example, where players x_3 and x_4 are in both sets).

If $\rho = ((x_{i_0}, x_{j_0}), \ldots, (x_{i_{r-1}}, x_{j_{r-1}}))$ is a rotation in a stable preference structure P, and x_{j_k} is a player in the target set of ρ, we denote by $x_{g(j_k)}$ the least favoured member of the set $(B_{j_k} \cup \{x_{i_{k-1}}\}) \backslash \{x_{i_k}\}$, and refer to $x_{g(j_k)}$ as x_{j_k}'s *next-worst bidder*.

We denote by P/ρ the preference structure obtained from P by deleting, for $0 \leqslant k \leqslant r-1$, all pairs $\{x_{j_k}, x_l\}$ such that x_{j_k} prefers his next-worst bidder $x_{g(j_k)}$ to x_l. One consequence is that x_i's worst bidder is removed from his list,

x_1 :(2)	$x_3\ x_2\ x_4\ x_5$	x_1 :(2)	$x_3\ x_2$
x_2 :(2)	$x_1\ x_4\ x_3\ x_5$	x_2 :(2)	$x_1\ x_3\ x_5$
x_3 :(2)	$x_8\ x_9\ x_1\ x_2\ x_4$	x_3 :(2)	$x_8\ x_1\ x_2$
x_4 :(2)	$x_5\ x_3\ x_9\ x_1\ x_8\ x_2$	x_4 :(2)	$x_5\ x_9\ x_8$
x_5 :(2)	$x_2\ x_1\ x_4\ x_9$	x_5 :(2)	$x_2\ x_4\ x_9$
x_6 :(2)	x_7	x_6 :(2)	x_7
x_7 :(1)	x_6	x_7 :(1)	x_6
x_8 :(1)	$x_4\ x_9\ x_3$	x_8 :(1)	$x_4\ x_9\ x_3$
x_9 :(1)	$x_5\ x_8\ x_4\ x_3$	x_9 :(1)	$x_5\ x_8\ x_4$
x_{10}:(1)		x_{10}:(1)	

Phase 1 preference structure P^1 Preference structure $P = P^1/\rho$

$$S(P) = \{(x_1,x_3), (x_1,x_2), (x_2,x_1), (x_2,x_3), (x_3,x_8), (x_3,x_1), (x_4,x_5), (x_4,x_9), (x_5,x_2), (x_5,x_4), (x_6,x_7), (x_7,x_6), (x_8,x_4), (x_9,x_5)\}$$

Fig. 3. The effect of rotation elimination.

```
P = P^1;
if Σ_i d_i is odd
    report instance unsolvable;
else
{
    while (there are no short lists in P) and
        (there is some long list in P)
    {
        find a rotation ρ in P;
        P = P/ρ;
    }
    if some list in P is short
        report instance unsolvable;
    else
        output P, which is a stable matching;
}
```

Fig. 4. Algorithm SF-Phase2.

and his next-worst bidder becomes his worst bidder as a result. The process of replacing P by P/ρ is referred to as *eliminating* rotation ρ. □

Example. For the example instance illustrated in Fig. 2, the effect of eliminating the rotation $\rho = ((x_4, x_3), (x_3, x_9), (x_5, x_1), (x_2, x_4))$ is shown in Fig. 3.

Note that there have been deletions from the preference lists of all of the players involved in the rotation.

In phase 2 of the algorithm the preference structure is reduced by successive elimination of rotations. The starting point is the phase 1 preference structure P^1, which we know is stable. Throughout the reduction, provided no player's preference list becomes short (i.e., contains fewer entries than his degree), the current preference structure will be shown to be stable, so that Lemma 3.3 applies throughout, and the reduction process can continue as long as some player has a long preference list. We will also show that, if a stable matching does exist for the instance, then there is always at least one such matching embedded in each of the stable preference structures that is generated.

Phase 2 of the algorithm is summarised in Fig. 4.

Lemma 3.4. *Let ρ be a rotation in a stable preference structure P. Then, if P/ρ contains no short list, P/ρ is itself a stable preference structure.*

Proof. If $P' = P/\rho$ contains no short list we need to show that the required three properties are satisfied by P'. For brevity, we denote the sets associated with $S(P')$ by A'_i and B'_i, of sizes a'_i and b'_i, respectively.

Property SPS1. By definition, $a_i' = d_i$, since P_i' is not a short list. It remains to show that $b_i' = d_i$. As observed earlier, pairs deleted from P on eliminating ρ are of the form $\{x_l, x_j\}$, where x_j is in the target set of ρ and x_j prefers his next-worst bidder to x_l. This applies to only one pair of $S(P)$, namely $\{x_{l(j)}, x_j\}$. Hence $x_{l(j)}$ is the only element of B_j that is not in B_j'. However, if x_i is x_j's new bidder, then x_i is in B_j' but not in B_j, and x_i is the only player with this property. Hence $b_j' = b_j = d_j$ as required.

Property SPS2. Because of the deletions specified when ρ is eliminated, the last entry in P_i' is either the same as in P_i or, if x_i is in the target set of ρ, it is his next-worst bidder. This player is in B_i', so the required property holds.

Property SPS3. Let $\{x_i, x_j\}$ be an acceptable pair that is not in P'. If $\{x_i, x_j\}$ is not even in P then the result is immediate. Otherwise $\{x_i, x_j\}$ must have been deleted because, say, x_i prefers his next-worst bidder to x_j. But this is his worst bidder in P', so the required property follows. Conversely, the only acceptable pairs deleted when ρ is eliminated are precisely pairs for which the required condition is satisfied. □

We are now in a position to prove the final lemma required to establish the correctness of algorithm `SF-Phase2`.

Lemma 3.5. *If there is a stable matching contained within a stable preference structure P, and if ρ is a rotation in P, then there is a stable matching contained within P/ρ.*

Proof. Let $\mathscr{M}$ be a stable matching contained within P, and suppose that $\rho = ((x_{i_0}, x_{j_0}), \ldots, (x_{i_{r-1}}, x_{j_{r-1}}))$ is a rotation in P.

Case (i): for some k $(0 \leqslant k \leqslant r-1)$, $\{x_{i_k}, x_{j_k}\} \notin \mathscr{M}$. We can assume, without loss of generality, that $\{x_{i_0}, x_{j_0}\} \notin \mathscr{M}$. Suppose that $\{x_{j_1}, x_l\} \in M$ for some x_l such that x_{j_1} prefers his next-worst bidder to x_l. Since $\{x_{i_0}, x_{j_0}\} \notin \mathscr{M}$, since x_{j_0} is a target of x_{i_0}, and since x_{i_0} has exactly d_{i_0} stable matches in $\mathscr{M}$, by Lemma 2.5 and the fact that P_{i_0} is a long list, we must have $\{x_{i_0}, x_k\} \in \mathscr{M}$ for some x_k who is not a target of x_{i_0}.

Subcase (ia): $\{x_{i_0}, x_{j_1}\} \in \mathscr{M}$. Since there are two non-bidders, x_l and x_{i_0}, who are matched with x_{j_1} in $\mathscr{M}$, there must be some bidder other than x_{i_1}, say x_m, who is not matched with x_{j_1} in $\mathscr{M}$. So x_{j_1} prefers x_m to x_l, one of his matches in $\mathscr{M}$, and, since x_m must have d_m matches in $\mathscr{M}$, he prefers x_{j_1}, one of his targets, to one of those matches. Hence $\{x_{j_1}, x_m\}$ blocks $\mathscr{M}$.

Subcase (ib): $\{x_{i_0}, x_{j_1}\} \notin \mathscr{M}$ and $\{x_{i_0}, x_k\} \in \mathscr{M}$ for some x_k lower in x_{i_0}'s list than x_{j_1}. Then x_{i_0} prefers x_{j_1} to one of his matches in $\mathscr{M}$, and prefers x_{i_0}, one of his targets, to x_l, one of his matches in $\mathscr{M}$. Hence $\{x_{i_0}, x_{j_1}\}$ blocks $\mathscr{M}$.

So all entries in P_{j_1} worse than x_{j_1}'s next-worst bidder can be deleted without deleting any pairs of $\mathscr{M}$. In so doing, $\{x_{i_1}, x_{j_1}\}$ is deleted, so this pair is not in $\mathscr{M}$, and the same argument can be repeated ($r-1$ times) to show that ρ can be eliminated without deleting any pairs of $\mathscr{M}$. Thus $\mathscr{M}$ is contained in P/ρ.

Case (ii): for all k $(0 \leqslant i \leqslant r-1)$, $\{x_{i_k}, x_{j_k}\} \in \mathscr{M}$. We show that if $\mathscr{M}'$ is obtained from $\mathscr{M}$ by replacing $\{x_{i_k}, x_{j_k}\}$ by $\{x_{i_k}, x_{j_{k+1}}\}$ for all k ($k+1$ taken modulo r), then $\mathscr{M}'$ is also a stable matching.

Firstly, $\mathscr{M}'$ is a matching, since each player has the same number of matches in $\mathscr{M}'$ as in $\mathscr{M}$.

Secondly, we have to show that $\mathscr{M}'$ is contained in P/ρ. None of these new pairs $\{x_{i_k}, x_{j_{k+1}}\}$ is deleted when ρ is eliminated. Suppose that some other pair in $\mathscr{M}$ is deleted. The only pairs deleted are of the form $\{x_{j_k}, x_l\}$, where x_{j_k} prefers his next-worst bidder to x_l. If $\{x_{j_k}, x_l\} \in \mathscr{M}$ for such an x_l, then x_l cannot be a bidder for x_{j_k}. Hence some bidder for x_{j_k} other than x_{i_k}, say x_r, is not matched in $\mathscr{M}$ with x_{j_k}, since both the number of bidders and the number of matches must be equal to the degree d_{j_k}. It follows that x_{j_k} prefers x_r to at least one of his matches in $\mathscr{M}$. Also, x_{j_k} is among the first d_r entries in x_r's list in P, so x_r prefers x_{j_k} to one of his matches in $\mathscr{M}$ (which must number d_r). Hence $\{x_{j_k}, x_r\}$ is a blocking pair for $\mathscr{M}$, a contradiction.

Finally, we have to show that $\mathscr{M}'$ is stable. If there is a blocking pair for $\mathscr{M}'$ then one or both members of the pair must have a match in $\mathscr{M}'$ who is less desirable than his worst match in $\mathscr{M}$, otherwise the pair blocks $\mathscr{M}$ also. The only players for whom this is the case are those in the bidder set of ρ, so suppose that the blocking pair is $\{x_{i_k}, x_m\}$ for some k and m. Furthermore, for x_{i_k} to have a match in $\mathscr{M}'$ that is worse than any of his matches in $\mathscr{M}$, it must be the case that x_{i_k} is matched in M with all of his targets in P, and that x_{i_k} prefers x_m to his next target $x_{j_{k+1}}$, but not to any of his existing targets. Player x_m cannot be x_{j_k}, because x_{j_k} prefers all the players in his preference list in P/ρ, and hence all his matches in $\mathscr{M}$, to x_{i_k}. The remaining possibility is that x_m lies in x_{i_k}'s preference list between his last target in P and $x_{j_{k+1}}$. But then the pair $\{x_{i_k}, x_m\}$ is not in P, and by property SPS3, this can only be because x_m prefers $x_{l(m)}$, and therefore all of his matches in $\mathscr{M}'$, to x_{i_k}. □

Preference structure P		Preference structure P/ρ'	
x_1 :(2)	$x_3\ x_2$	x_1 :(2)	$x_3\ x_2$
x_2 :(2)	$x_1\ x_3\ x_5$	x_2 :(2)	$x_1\ x_3$
x_3 :(2)	$x_8\ x_1\ x_2$	x_3 :(2)	$x_1\ x_2$
x_4 :(2)	$x_5\ x_9\ x_8$	x_4 :(2)	$x_5\ x_8$
x_5 :(2)	$x_2\ x_4\ x_9$	x_5 :(2)	$x_4\ x_9$
x_6 :(2)	x_7	x_6 :(2)	x_7
x_7 :(1)	x_6	x_7 :(1)	x_6
x_8 :(1)	$x_4\ x_9\ x_3$	x_8 :(1)	x_4
x_9 :(1)	$x_5\ x_8\ x_4$	x_9 :(1)	x_5
x_{10}:(1)		x_{10}:(1)	

Fig. 5. Elimination of a further rotation to give a stable matching.

x_1 :(2)	$x_2\ x_4\ x_3$
x_2 :(2)	$x_3\ x_5\ x_1$
x_3 :(2)	$x_1\ x_6\ x_2$
x_4 :(2)	$x_5\ x_1\ x_6$
x_5 :(2)	$x_6\ x_2\ x_4$
x_6 :(2)	$x_4\ x_3\ x_5$

Preference structure $P = P^1$

$$S(P) = \{(x_1,x_2), (x_1,x_4), (x_2,x_3), (x_2,x_5), (x_3,x_1), (x_3,x_6), (x_4,x_5), (x_4,x_1), (x_5,x_6), (x_5,x_2), (x_6,x_4), (x_6,x_3)\}$$

Fig. 6. An SF instance for which no stable matching exists.

Theorem 3.1. *Algorithm* `SF-Phase2` *correctly identifies whether a stable matching exists for an SF instance, and returns such a matching when one does exist.*

Proof. Corollary 2.2 establishes that no stable matching can exist if $\sum d_i$ is odd. Otherwise, Lemmas 3.3–3.5 justify the successive elimination of rotations. Lemma 3.5 shows that, if a short list occurs, then there cannot be a stable matching, whereas if there is a stable matching then one such matching is returned by the algorithm. □

As with the classical SR problem [5], in cases where more than one stable matching exists, the particular stable matching returned by `SF-Phase2` for an instance of SF depends on the set of rotations eliminated.

Example. Recall that Fig. 3 shows the preference structure after the elimination of one rotation for our example SF instance. Fig. 5 displays this structure P again. There is only one rotation in P, namely $\rho' = ((x_5, x_2), (x_4, x_9), (x_3, x_8))$. When this rotation is eliminated we obtain the new stable preference structure P/ρ', shown on the right in Fig. 5, which is a stable matching.

Example. As a second example, this time one for which no stable matching exists, consider the preference structure P shown in Fig. 6. In this case `SF-Phase1` results in no deletions from the structure. In `SF-Phase2`, there are two rotations in P, namely $\rho_1 = ((x_1, x_3), (x_2, x_1), (x_3, x_2))$ and $\rho_2 = ((x_4, x_6), (x_5, x_4), (x_6, x_5))$. Elimination of ρ_1 results in short lists for x_1, x_2 and x_3, so there can be no stable matching. (Of course a similar conclusion follows if we eliminate ρ_2 instead, as this yields short lists for x_4, x_5 and x_6.)

4. Implementation and analysis

We now show that the SF algorithm can be implemented in such a way as to have O(m) worst-case complexity for an instance involving preference lists with total combined length m.

In phase 1 of the algorithm, deletion of a pair $\{x_i, x_j\}$ can be achieved in constant time by holding a separate ranking array for each player such that, in the array for x_i, position j holds the index of the position that x_j occupies in P_i. This means that, when player x_j is deleted from the list of player x_i, x_i can be deleted from x_j's list using one look-up of the ranking array. This second part of the deletion is achieved in constant time if the preference lists are stored in an indexed, doubly linked structure. The number of deletions is bounded by m, the total length of the preference lists, as is the number of bids. At the point at which a player x_i receives a bid for the c_ith time it is simple to ascertain which element in P_i represents the least favoured of his c_i bidders. Thereafter, when a player x_i receives a further bid, the c_ith ranked of all of his bidders can be found by stepping backwards in P_i from the last entry to the next bidder found. The total number of such backward steps is bounded by m, so phase 1 certainly has worst-case complexity $O(m)$.

For phase 2 of the algorithm a stack is used to house a path traced out in the directed graph $\mathscr{D}(P)$. When a node is reached that is already on the stack, a rotation has been found and it may be recovered by popping the stack as far as the first occurrence of that node. The rotation may then be eliminated by deleting the necessary pairs, and each deletion is a constant time operation, as has been noted above. Each successive rotation search begins from the end of the previous tail, so each pair encountered is a new pair. This ensures that the same, potentially long, tail will not be traversed more than once. Since the stack must be empty at the end of the execution of the algorithm (it only ever contains players who have long lists) the number of push and pop operations is the same. Further, at least one pair is deleted every time a pop operation is performed, so the number of pop (and hence push) operations cannot exceed m, and since every operation associated with finding a rotation and deleting a pair can be achieved in constant time, it follows that the whole algorithm is $O(m)$.

Finally, since the SM problem is a special case of the SF problem, and since there is an $\Omega(m)$ lower bound for SM [9], the SF algorithm is asymptotically optimal.

5. Conclusion and open questions

In this paper we have introduced the SF problem, and described our motivation for studying it. We have established that there is an $O(m)$ algorithm for determining if an instance of SF admits a stable matching, and if it does, to find one such matching. However, a number of open questions remain, including the following.

Open question 1. For an instance of SF in which the preference lists are complete, what is the smallest possible size of a stable matching (expressed in terms of the size of the instance, n, and the capacities)?

Open question 2. Is there any analogue of the "medians" result for the SR problem, namely that the so-called median of any 3 stable matchings is itself a stable matching [4]?

Open question 3. In an instance of SF in which players are allowed to express indifference between two or more other players, there are a number of possibilities for defining a blocking pair, leading to the notions of weak stability, strong stability and super-stability [6,7]. For weak stability SF is NP-complete, as a consequence of the corresponding result for SR [10]. For super-stability, Scott [12] has described a polynomial-time algorithm that builds on the algorithm presented in this paper. It is open as to whether there exists a polynomial-time algorithm to solve SF in the case of strong stability.

Acknowledgement

The authors gratefully acknowledge the helpful comments of the referees, which led to substantial improvements on an earlier version of the paper.

References

[1] M. Baïou, M. Balinski, Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry), Discrete Appl. Math. 101 (2000) 1–12.
[2] D. Gale, L.S. Shapley, College admissions and the stability of marriage, Amer. Math. Monthly 69 (1962) 9–15.
[3] D. Gusfield, The structure of the stable roommate problem—efficient representation and enumeration of all stable assignments, SIAM J. Comput. 17 (4) (1988) 742–769.
[4] D. Gusfield, R.W. Irving, The Stable Marriage Problem: Structure and Algorithms, MIT Press, Cambridge, 1989.
[5] R.W. Irving, An efficient algorithm for the "stable roommates" problem, J. Algorithms 6 (1985) 577–595.
[6] R.W. Irving, Stable marriage and indifference, Discrete Appl. Math. 48 (1994) 261–272.
[7] R.W. Irving, D.F. Manlove, The stable roommates problem with ties, J. Algorithms 43 (2002) 85–105.

[8] D.E. Knuth, Stable marriage and its relation to other combinatorial problems, in: CRM Proceedings and Lecture Notes, vol. 10, American Mathematical Society, Providence, RI, 1997.
[9] C. Ng, D.S. Hirschberg, Lower bounds for the stable marriage problem and its variants, SIAM J. Comput. 19 (1990) 71–77.
[10] E. Ronn, NP-complete stable matching problems, J. Algorithms 11 (1990) 285–304.
[11] A.E. Roth, M.A.O. Sotomayor, Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis, Cambridge University Press, Cambridge, 1990.
[12] S. Scott, A study of stable marriage problems with ties, Ph.D. Thesis, University of Glasgow, Department of Computing Science, 2005.