
http://eprints.gla.ac.uk/25729/

Deposited on: 07 April 2010
Keeping partners together:  
Algorithmic results for the Hospitals / Residents problem with couples*  

Eric J. McDermid and David F. Manlove  

Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK  
E-mail: {mcdermid,davidm}@dcs.gla.ac.uk.

Abstract  
The Hospitals / Residents problem with Couples (HRC) is a generalisation of the classical Hospitals / Residents problem (HR) that is important in practical applications because it models the case where couples submit joint preference lists over pairs of hospitals \( (h_i, h_j) \). We consider a natural restriction of HRC in which the members of a couple have individual preference lists over hospitals, and the joint preference list of the couple is consistent with these individual lists in a precise sense. We give an appropriate stability definition and show that, in this context, the problem of deciding whether a stable matching exists is NP-complete, even if each resident’s preference list has length at most 3 and each hospital has capacity at most 2. However, with respect to classical (Gale-Shapley) stability, we give a linear-time algorithm to find a stable matching or report that none exists, regardless of the preference list lengths or the hospital capacities. Finally, for an alternative formulation of our restriction of HRC, which we call the Hospitals / Residents problem with Sizes (HRS), we give a linear-time algorithm that always finds a stable matching for the case that hospital preference lists are of length at most 2, and where hospital capacities can be arbitrary.

1 Introduction  
An instance \( I \) of the classical Hospitals / Residents problem (HR) [6] involves two sets, namely a set \( R = \{r_1, \ldots, r_n\} \) of residents and a set \( H = \{h_1, \ldots, h_m\} \) of hospitals. Each resident in \( R \) seeks to be assigned to a hospital, whilst each hospital \( h_j \in H \) has a capacity \( c_j \in Z^+ \) indicating the maximum number of residents who could be assigned to \( h_j \). Each resident \( r_i \in R \) ranks a subset of \( H \), his acceptable hospitals, in strict order of preference, and each hospital \( h_j \in H \) ranks, again in strict order, those residents who find \( h_j \) acceptable. A solution of \( I \) is a matching (i.e., an assignment of mutually acceptable (resident, hospital) pairs such that no resident is assigned more than one hospital, and no hospital is assigned more residents than its capacity) that is stable [6, 9]. A matching is stable if it admits no blocking pair. Informally, a blocking pair of \( M \) is a resident and hospital who would prefer to be assigned to one another than remain with their allocations in \( M \). It is known that every instance of HR admits a stable matching, and that such a matching can be found in linear time using the extended Gale-Shapley algorithm [6];[9, Section 1.6].

*Supported by Engineering and Physical Sciences Research Council grant EP/E011993/1.
HR is a many-one extension of the classical Stable Marriage problem [6], so called because of its widespread application to centralised automated matching schemes that allocate graduating medical students (residents) to hospital posts. In particular the National Resident Matching Program (NRMP) in the USA [21], the Canadian Resident Matching Service [22] and the Scottish Foundation Allocation Scheme (SFAS) [10, 23] all essentially incorporate extensions of the Gale-Shapley algorithm for HR.

Couples

In the above practical applications, the existence of couples who wish to be located at the same hospital, or at hospitals close to one another, gives rise to an important variant of HR called the Hospitals / Residents problem with Couples (HRC) [18, 17, 5, 2, 12, 13, 14]. The study of this problem was motivated by the fact that, over the years, participation in the NRMP was observed to decrease, possibly as a result of the original algorithm being unable to accommodate the complicated preference structure of couples [9, p.4];[20, p.7].

An instance of HRC involves both single residents and couples (pairs of residents) such that each resident belongs to at most one couple. Each couple \((r_i, r_j)\) has a preference list over pairs of hospitals \((h_k, h_l)\), representing the assignment of \(r_i\) to \(h_k\) and of \(r_j\) to \(h_l\). Ronn [17] (see also [9, Section 1.6.6]) described a stability criterion for a matching in HRC that is a natural generalisation of the analogous concept in the HR context (we define this stability concept formally in Section 2). Roth [18] showed that an HRC instance need not admit a stable matching, whilst Ronn proved that the problem of deciding whether an HRC instance admits a stable matching is NP-complete, even if there are no single residents and each hospital has capacity 1 [17].

Consistent couples

In this paper we consider a natural restriction of HRC in which each member of a given couple \((r_i, r_j)\) has an individual preference list over a subset of hospitals, and the joint preference list of the couple is consistent with the individual preferences of \(r_i\) and \(r_j\) in a precise sense. That is, \((r_i, r_j)\) ranks distinct pairs of hospitals in order of preference, such that \((h_p, h_q)\) precedes \((h_r, h_s)\) on this list implies that (i) either \(r_i\) prefers \(h_p\) to \(h_r\) or \(h_p = h_r\), and (ii) either \(r_j\) prefers \(h_q\) to \(h_s\), or \(h_q = h_s\). We refer to this restriction of HRC as the Hospitals / Residents problem with Consistent Couples (HRCC).

Thus HRCC models a situation in which the members of each couple can agree to construct a joint preference list from their individual preferences consistently, in the sense that if a couple jointly prefers \((h_p, h_q)\) to \((h_r, h_s)\), then when comparing \(h_p\) to \(h_r\), \(r_i\) would be no worse off, and similarly when comparing \(h_q\) to \(h_s\), \(r_j\) would be no worse off. This includes the case where both members of a given couple have identical individual preference lists, with the intended outcome being that they are either matched to the same hospital or not matched at all.

HRCC does not seem to have been studied previously in the literature from an algorithmic point of view. In this paper we show that an instance \(I\) of HRCC need not admit a stable matching, and that the problem of deciding whether \(I\) admits a stable matching is NP-complete. This result holds even if the length of each resident’s individual list and the length of each couple’s joint list is at most 3, and the capacity of each hospital is at most 2, thus providing another highly restricted version of HRC that remains NP-complete, in addition to the case considered by Ronn [17]. This restriction is important from a practical viewpoint, because in many applications the preference lists on one side tend to be short (for example in the context of SFAS, residents are asked to rank up to 6 hospitals in order of preference).
By contrast, we also give a linear-time algorithm to find a stable matching or report that none exists, for the case that stability is defined with respect to classical (Gale-Shapley) stability (that is, each member of a couple can form a blocking pair with a hospital without regard to the other member of the couple). This version of stability can be motivated in the HRCC context as follows. Suppose that a given couple \((r_i, r_j)\) is given the joint assignment \((h_{r_i}, h_{r_j})\) by a matching algorithm. Now suppose that \(r_i\) prefers some hospital \(h_p\) to \(h_{r_i}\), whilst the joint assignment \((h_p, h_{r_j})\) is not acceptable to the couple for whatever reason. The previously stated acceptance of the couple to supply a joint (consistent) preference list could be overridden in practice if \(r_i\) has an overarching desire to be allocated to \(h_p\) as opposed to \(h_{r_i}\) (where \(h_p\) may be far away from \(h_{r_i}\)). In reality this could mean that either \(r_j\) moves with \(r_i\) to remain geographically close, and attempts to make an arrangement with \(h_p\) (or a hospital nearby) outside of the matching scheme, or \(r_j\) changes career, or indeed the couple even split up. In the spirit of “keeping partners together”, as indicated by the title of this paper, this is a situation that we seek to avoid, thus motivating this stronger form of stability in the context of HRCC. Hence we obtain a natural restriction of HRC that, unlike the general problem, is solvable in polynomial time. Our algorithm does not make any assumptions on the lengths of the preference lists or on the hospital capacities.

We remark that a matching that satisfies classical stability in the context of HRCC is stable with respect to the criteria defined earlier by Roth and Ronn [19, 17] for HRC (see Section 2 for a formal definition of this stability criterion). Note that the converse, however, is not true in general. Our algorithm for HRCC under classical stability helps to narrow the search for the boundary between polynomial time solvable and NP-complete variants of HRC. In particular, HRCC under classical stability is the most general restriction of HRC that we are aware of that remains solvable in polynomial time.

**Hospitals / Residents problem with Sizes**

As alluded to above, a special case of HRCC arises when each couple \((r_i, r_j)\) satisfies the property that the individual preference lists of \(r_i\) and \(r_j\) are identical, and the joint preference list of \((r_i, r_j)\) satisfies the property that \(h_p = h_q\) for any element \((h_p, h_q)\) on this list. Thus \(r_i\) and \(r_j\) wish to be either assigned to the same hospital, or both be unassigned. We refer to this restriction of HRIC as the **Hospitals / Residents problem with Inseparable Couples (HRIC)**.

Let \(I\) be an instance of HRIC and let \((r_i, r_j)\) be a couple in \(I\). Given the structure of \((r_i, r_j)’s\) preference list, it is natural to replace \((r_i, r_j)\) by a single entity \(C_{i,j}\) whose preference list is obtained from that of \((r_i, r_j)\) by replacing each occurrence of \((h_k, h_k)\) by \(h_k\). Thus each single resident occupies one post at a given hospital, whilst each couple occupies two posts. This suggests a natural generalisation of HRIC to the case where each resident \(r_i \in R\) has a size \(s_i \in \mathbb{Z}^+\), indicating the number of posts that \(r_i\) occupies at any hospital. Hospitals will now rank residents of any size (including couples) as a single entity. We refer to this variant of HRC as the **Hospitals / Residents problem with Sizes (HRS)**.

A formal definition of HRS is given in Section 2, in which we formulate an appropriate notion of stability in the HRS context. With this stability definition we later prove that, given an HRS instance where the size of each resident is at most 2 and the capacity of each hospital is at most 2, the problem of deciding whether a stable matching exists is NP-complete, even if the length of each preference list is at most 3. We also show that the restriction of HRS in which each resident has size at most 2 is reducible to HRCC (essentially each resident of size 2 becomes a couple), thus implying the aforementioned NP-completeness result for HRCC.
However by contrast we also prove that, given an instance of HRS in which the length of each hospital’s preference list is at most 2, a stable matching always exists and can be found in linear time. The result holds for arbitrary resident sizes and hospital capacities. This result therefore indicates a boundary between the polynomial-time solvability and NP-completeness of HRS with respect to the length of a hospital’s preference list.

Related work

A version of HRS, called the Unsplittable Stable Marriage problem, was studied previously by Dean et al. [4]; their version differs from ours in that they permit a hospital $h_j$’s capacity to be exceeded by the assignment of a couple to $h_j$. Dean et al. formulate the problem in terms of assigning jobs (residents) with integral sizes to machines (hospitals) with capacities. They provide a polynomial-time integral variant of the Gale-Shapley algorithm that finds a stable matching in which each machine is congested by at most the processing time of the largest job. In the analogous HRS setting, their algorithm finds a stable matching in which the capacity of each hospital is oversubscribed by at most the size of the largest resident. Until now, the complexity of determining the existence of a stable matching in which none of the hospitals’ capacity constraints are exceeded was an open problem.

Organisation of the paper

The remainder of this paper is organised as follows. In Section 2 we give a formal definition of each of HRS and HRCC. We also show that a restricted version of the former problem can be reduced to the latter, and that an instance of either problem need not admit a stable matching. In Section 3 we prove that the problem of determining whether an HRS instance admits a stable matching is NP-complete, even if the size and capacity of each resident and hospital is at most 2 respectively, and the preference list of each resident and hospital is at most 3. Together with the reduction given in Section 2, this also establishes the NP-completeness of determining whether an HRCC instance admits a stable matching. In Section 4, we consider HRCC under classical (Gale-Shapley) stability, and show that the problem of finding, given an HRCC instance, a matching that satisfies this form of stability, or reporting that none exists, is solvable in linear time. In Section 5 we revisit HRS and consider the case where the length of each hospital’s preference list is at most 2. Given an instance of this restricted version of the problem, we give a linear-time algorithm for finding a stable matching.

2 Formal definitions of HRS and HRCC

We firstly give a formal definition of the Hospitals / Residents problem with Sizes (HRS). An instance $I$ of this problem is defined in the same way as an instance of HR (as defined in Section 1) except that each resident $r_i \in R$ has a size $s_i \in \mathbb{Z}^+$. An assignment $M$ in $I$ is a set of (resident,hospital) pairs such that $(r_i, h_j) \in M$ only if $r_i$ and $h_j$ find each other acceptable. For $r_i \in R$ we denote $\{h_j \in H : (r_i, h_j) \in M\}$ by $M(r_i)$, for $h_j \in H$ we denote $\{r_i \in R : (r_i, h_j) \in M\}$ by $M(h_j)$, and for $h_j \in H$ we denote $\sum \{s_i : r_i \in M(h_j)\}$ by $O_M^j$ and refer to this as the occupancy of $h_j$ in $M$. We say that $h_j$ is undersubscribed if $O_M^j < c_j$.

A matching is an assignment $M$ such that $|M(r_i)| \leq 1$ for each $r_i \in R$ and $O_M^j \leq c_j$ for each $h_j \in H$. In other words, each resident is assigned to at most one hospital, and the sum of the sizes of the residents assigned to a hospital does not exceed its capacity. Given a matching $M$ in which a resident $r_i$ is matched to a hospital $h_j$, with a slight abuse
A matching pair \((h_i, h_j)\) of acceptable hospitals. Each entry in this list is an ordered number of notation we let \(M(r_i)\) denote \(h_j\). A pair \((r_i, h_j)\) blocks a matching \(M\), or is a blocking pair for \(M\), if

1. \(r_i\) is unmatched, or \(r_i\) prefers \(h_j\) to \(M(r_i)\), and

2. \(O_j^M + s_i - \sum_{p=1}^{t} s_{kp} \leq c_j\).

The definition implies that \(h_j\) could participate in a blocking pair with \(r_i\) if (i) either \(h_j\) currently has room for \(r_i\), or (ii) \(h_j\) can make room for \(r_i\) by rejecting a set of residents it finds worse than \(r_i\). A matching is stable if it admits no blocking pair.

We assume without loss of generality that, for each \(r_i \in R\) and for each hospital \(h_j\) on \(r_i\)’s preference list, \(s_i \leq c_j\), for otherwise \((r_i, h_j)\) could never belong to a stable matching, nor could \((r_i, h_j)\) form a blocking pair.

We firstly observe that clearly HR is the special case of HRS in which \(s_i = 1\) for each \(r_i \in R\). The blocking pair definition for HR (which gives rise to the classical stability definition) can then be deduced from that for HRS by interpreting Condition 2 as follows: either \(h_j\) is undersubscribed or prefers \(r_i\) to some resident in \(M(h_j)\).

We next observe that, in contrast to HR, an HRS instance may not admit a stable matching. An example instance \(I\) that illustrates this is shown in Figure 1 (in this figure, and throughout the paper, sizes and capacities are written next to the residents and hospitals, respectively). Suppose for a contradiction that \(I\) admits a stable matching \(M\). If \((r_3, h_1) \in M\), then \((r_1, h_2) \in M\) or else \((r_1, h_1)\) blocks \(M\). Hence \((r_2, h_2)\) blocks \(M\), a contradiction. Hence \((r_1, h_1) \notin M\). Then \((r_2, h_1) \in M\), or else \((r_2, h_1)\) blocks \(M\). Hence \((r_1, h_2) \in M\) or else \((r_1, h_2)\) blocks \(M\). Hence \((r_3, h_1)\) blocks \(M\), a contradiction.

Our third observation is that the restriction of HRS where each resident has size at most 2 is reducible to the Hospitals / Residents problem with Consistent Couples (HRCC), which is a special case of the Hospitals / Residents problem with Couples (HRC). We now demonstrate this, but first we give a formal definition of each of HRC and HRCC.

An instance \(I\) of HRC involves a set \(R = \{r_1, \ldots, r_n\}\) of residents, a set \(H = \{h_1, \ldots, h_m\}\) of hospitals, and a set \(C\) of couples, i.e., ordered pairs of residents such that each resident appears in at most one pair. As in the HR case, each hospital \(h_j \in H\) has a capacity \(c_j \in \mathbb{Z}^+\).

Each single resident \(r_i \in R\) (i.e., a resident who does not belong to a couple) submits a strict preference list of acceptable hospitals. Each couple \((r_i, r_j)\) submits a joint (strict) preference list over pairs of acceptable hospitals. Each entry in this list is an ordered pair \((h_k, h_l)\) of (not necessarily distinct) hospitals representing the assignment of \(r_i\) to \(h_k\) and of \(r_j\) to \(h_l\). Finally, each hospital \(h_j \in H\) ranks those residents (whether single or a member of a couple) who find \(h_j\) acceptable in strict order of preference.

In this context, the definition of a matching \(M\) is the same as in the classical HR setting, with the additional requirement that, for each couple \((r_i, r_j)\), if \((r_i, h_k) \in M\) and \((r_j, h_l) \in M\) then the pair \((h_k, h_l)\) must appear on the joint preference list of that couple. A matching \(M\) is unstable if at least one of the following holds:

<table>
<thead>
<tr>
<th>1:</th>
<th>2:</th>
<th>1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(r_1: h_2\ h_1)</td>
<td>(h_1: r_1\ r_3\ r_2)</td>
<td>(r_2: h_1\ h_2)</td>
</tr>
<tr>
<td>(r_3: h_1)</td>
<td>(1: h_2\ r_2\ r_1)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: An HRS instance for which no stable matching exists.
1. The matching is blocked by a hospital \( h_j \) and a single resident \( r_i \), as in the classical HR problem.

2. The matching is blocked by a hospital \( h_k \) and a resident \( r_i \) who is coupled, say with \( r_j \); that is, \((r_i, r_j)\) prefers \((h_k, M(r_j))\) to \((M(r_i), M(r_j))\), and \( h_k \) is either undersubscribed in \( M \) or prefers \( r_i \) to some member of \( M(h_k) \setminus \{r_j\} \).

3. The matching is blocked by a couple \((r_j, r_j)\) and (not necessarily distinct) hospitals \( h_k \neq M(r_i), h_l \neq M(r_j) \); that is, \((r_i, r_j)\) prefers the joint assignment \((h_k, h_l)\) to \((M(r_i), M(r_j))\), and either
   
   (a) \( h_k \neq h_l \), and \( h_k \) (respectively \( h_l \)) is either undersubscribed in \( M \) or prefers \( r_i \) (respectively \( r_j \)) to at least one of its assigned residents in \( M \); or
   
   (b) \( h_k = h_l \), and \( h_k \) has at least two free posts in \( M \), i.e., \( c_k - |M(h_k)| \geq 2 \); or
   
   (c) \( h_k = h_l \), and \( h_k \) has one free post in \( M \), i.e., \( c_k - |M(h_k)| = 1 \), and \( h_k \) prefers at least one of \( r_i, r_j \) to some member of \( M(h_k) \); or
   
   (d) \( h_k = h_l, h_k \) is full in \( M \), \( h_k \) prefers \( r_i \) to some \( r_s \in M(h_k) \), and \( h_k \) prefers \( r_j \) to some \( r_t \in M(h_k) \setminus \{r_s\} \).

The above stability definition for HRC extends that given in [9, Section 1.6.6], in order to deal with the case that \( h_k = h_l \), given a couple \((r_i, r_j)\) who prefer \((h_k, h_l)\) to \((M(r_i), M(r_j))\). As far as we are aware, this possibility does not appear to have been covered adequately by previous stability definitions for HRC in the literature [18, 9, 17, 5, 2, 12, 13, 14].

HRCC is the special case of HRC in which each resident (i.e., whether single or a member of a couple) ranks a subset of \( H \) in strict order of preference. Each couple \((r_i, r_j)\) ranks a subset of \( H \times H \) in strict order, subject to the constraint that this joint preference list be consistent with the individual preference lists of \( r_i \) and \( r_j \). That is, \((r_i, r_j)\) prefers \((h_p, h_q)\) to \((h_r, h_s)\) only if (i) either \( r_i \) prefers \( h_p \) to \( h_r \) or \( h_p = h_r \), and (ii) either \( r_j \) prefers \( h_q \) to \( h_s \) or \( h_q = h_s \). We now show that the restriction of HRS in which each resident has size at most 2 is polynomially reducible to HRCC.

**Lemma 2.1.** The restriction of HRS in which each resident has size at most 2 can be reduced in polynomial time to HRCC.

**Proof.** Given an instance \( I \) of the above version of HRS, construct an instance \( I' \) of HRCC in the following way. For each resident \( r_i \) of size 2, create a couple \((r_{i,1}, r_{i,2})\) in \( I' \). Suppose the preference list of \( r_i \) in \( I \) is \( h_1, h_2, \ldots, h_t \). Assign to each of \( r_{i,1} \) and \( r_{i,2} \) an individual list equal to that of \( r_i \). Let the joint preference list of \((r_{i,1}, r_{i,2})\) in \( I' \) be \((h_1, h_1), (h_2, h_2), \ldots, (h_t, h_t)\) – this is clearly consistent with the lists of \( r_{i,1} \) and \( r_{i,2} \).

For each hospital \( h_j \) that finds \( r_i \) acceptable in \( I \), replace the entry \( r_i \) on \( h_j \)'s preference list in \( I' \) with \( r_{i,1} \) and \( r_{i,2} \) in arbitrary order. Leave all residents of size 1 the same in both \( I \) and \( I' \). This ends the transformation. We claim that a stable matching exists for \( I' \) if and only if one exists for \( I \).

Suppose a stable matching \( M \) exists for \( I \). Then, construct a stable matching \( M' \) for \( I' \) in the following way. If \((r_i, h_j)\) is in \( M \), place \((r_i, h_j)\) into \( M' \) if \( r_i \) has size 1, else place \((r_{i,1}, h_j)\) and \((r_{i,2}, h_j)\) into \( M' \). Notice that the capacities of the hospitals are preserved in the reduction, and also that if a hospital \( h_j \) has an occupancy of \( t \) in \( M \), then \( h_j \) is assigned \( t \) residents in \( M' \). Suppose a blocking pair exists for \( M' \) in \( I' \). Then, the blocking pair must take the form of Rule 1 or 3 above, as Rule 2 is impossible by the special nature of the couple’s preference lists. If there is a blocking pair \((r_i, h_j)\) by Rule 1, \( M \) surely also had the same blocking pair in \( I \). If instead \( M' \) is blocked by Rule 3, then it must be
because a couple \((r_{i,1}, r_{i,2})\) block with the pair \((h_j, h_j)\) in \(I'\). But then resident \(r_i\) of size 2 in \(I\) must also block with hospital \(h_j\) in \(M\).

Conversely suppose a stable matching \(M'\) exists for \(I'\). Then, construct a stable matching \(M\) for \(I\) in the following way. If \((r_i, h_j)\) is in \(M'\), place \((r_i, h_j)\) into \(M\), if \(r_i\) has size 1 in \(I\), else if \((r_{i,1}, h_j)\) and \((r_{i,2}, h_j)\) are in \(M'\), place \((r_{i,1}, h_j)\) and \((r_{i,2}, h_j)\) into \(M\). By the nature of the preference lists, \(r_{i,1}\) and \(r_{i,2}\) are always assigned the same hospital. Suppose \((r_i, h_j)\) blocks \(M\) in \(I\). Then, by an argument similar to the above, \((r_i, h_j)\) must have blocked \(M'\) in \(I'\) if \(r_i\) has size 1, otherwise the pair \((r_{i,1}, r_{i,2})\) must have blocked \(M'\) in \(I'\) with \((h_j, h_j)\).

It follows immediately from Figure 1 and Lemma 2.1 that an HRCC instance need not admit a stable matching.

3 NP-completeness of HRS and HRCC

This section describes the polynomial-time reduction that establishes NP-completeness for the problem of deciding whether a stable matching exists, given an HRS instance where the sizes and capacities of the residents and the hospitals respectively is at most 2, and the length of each preference list is at most 3. This reduction begins from a problem we refer to as \((3,3,3)\)-COM-SMTI. In order to define this problem, we make the following preliminary definitions. The Stable Marriage problem with Incomplete lists (SMI) is the restriction of HR in which each hospital has capacity 1. The Stable Marriage problem with Ties and Incomplete lists (SMTI) is a generalisation of SMI in which preference lists can include ties. A matching \(M\) in an instance \(I\) of SMTI is stable if there is no man-woman pair, each of whom is either unmatched in \(M\) and finds the other acceptable, or prefers the other to his/her partner in \(M\). We then define \((3,3,3)\)-COM-SMTI to be the problem of deciding whether a complete stable matching exists (i.e., a stable matching that matches everybody), given an instance of SMTI in which each preference list is of length at most 3, every woman’s preference list is strictly ordered, and each man’s preference list is either strictly ordered or is a tie of length 2 (these conditions holding simultaneously). Using a modification of a reduction appearing in [11], we may deduce the following result, whose proof appears in the Appendix.

Theorem 3.1. \((3,3,3)\)-COM-SMTI is NP-complete.

Given an instance \(I\) of \((3,3,3)\)-COM-SMTI with \(n\) men \(m_1, m_2, \ldots, m_n\) and \(n\) women \(w_1, w_2, \ldots, w_n\), we create an instance \(I'\) of HRS, whose residents and hospitals are constructed as follows. Firstly, a hospital \(h_j\) is created for each woman \(w_i\) in \(I\).

Next, for each man \(m_i\) in \(I\) with a preference list consisting of a two-way tie \((w_k, w_l)\) where \(k < l\), create eight residents \(\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}, r_{i,1,1}, r_{i,1,2}, r_{i,2,1}, r_{i,2,2}\}\) and six hospitals \(\{h_{i,1}, h_{i,2}, h_{i,3,1}, h_{i,3,2}, h_{i,3,3}\}\). The preference lists, sizes and capacities of these eight residents and six hospitals are shown in Figure 2. For each man \(m_i\) in \(I\) with a strictly ordered preference list, create three residents \(\{r_{i,1}, r_{i,2}, r_{i,3}\}\) and two hospitals \(\{h_{i,1}, h_{i,2}\}\). The preference lists, sizes and capacities of these three residents and two hospitals are also shown in Figure 2.

Finally, for each hospital \(h_{ij}\) created from a woman \(w_k\) with preference list \(m_{i,1}, \ldots, m_{i,t_{ij}}\), set the preference list of \(h_{ij}\) to initially be equal to \(m_{i,1}, \ldots, m_{i,t_{ij}}\), temporarily placing “men” on \(h_{ij}\)’s preference list. Now, suppose that \(w_k\) finds some man \(m_j\) acceptable. If \(m_j\)’s preference list is strictly ordered in \(I\), replace \(m_j\) on \(h_{ij}\)’s preference list with \(r_{j,1}\). If \(m_j\)’s preference list is not strictly ordered, his preference list consists of a two-way tie, say, \((w_1, w_k)\). If \(t < k\), replace \(m_j\) with \(r_{j,1}\) on \(h_{ij}\)’s preference list, else, replace \(m_j\) with \(r_{j,2}\) on \(h_{ij}\)’s preference list. Set the capacity of \(h_{ij}\) to be 2.
This ends the reduction. Clearly, it is computable in polynomial time. We now argue that it is correct by the following sequence of lemmas, each of which states a property of any stable matching $M'$ in $I'$.

**Lemma 3.2.** Let $m_s$ be a man with a strictly ordered preference list in $I$, and let $m_r$ be a man with a preference list consisting of a two-way tie in $I$. Then, every resident in the set \{$r_s, r_{i,1}, r_{i,2}$\} is matched to some hospital in $M'$, and that hospital is not the last entry on his preference list.

**Proof.** Suppose that $r_s$ is unmatched in $M'$. Then, $r_{s_{s,1}}$ must be matched to $h_{s_{s,1}}$, to prevent $r_s$ from forming a blocking pair with $h_{s_{s,1}}$. Resident $r_{s_{s,2}}$ must be matched to $h_{s_{s,1}}$ as well, for otherwise he forms a blocking pair with $h_{s_{s,1}}$. But this implies $(r_{s_{s,1}}, h_{s_{s,2}})$ is a blocking pair for $M'$.

Suppose instead that $r_s$ is matched to $h_{s_{s,1}}$. Then, neither $r_{s_{s,1}}$ nor $r_{s_{s,2}}$ is matched to $h_{s_{s,1}}$, else its capacity would be exceeded. So, if $r_{s_{s,1}}$ is matched to $h_{s_{s,2}}$, $r_{s_{s,2}}$ is unmatched, and forms a blocking pair with $h_{s_{s,2}}$. If, instead, $r_{s_{s,1}}$ is matched to $h_{s_{s,2}}$, then $r_{s_{s,1}}$ is unmatched, and forms a blocking pair with $h_{s_{s,1}}$. Clearly, if neither $r_{s_{s,1}}$ nor $r_{s_{s,2}}$ is matched to $h_{s_{s,2}}$, they form blocking pairs with $h_{s_{s,2}}$. This exhausts every possibility. It follows that if $r_s$ is unmatched in $M'$ or is matched to the last hospital on his preference list, a blocking pair cannot be avoided.

The same argument holds for $r_{i,1}$ by substituting $h_{i_{o,1}}$ and $h_{i_{o,2}}$ for $h_{s_{s,1}}$ and $h_{s_{s,2}}$, respectively, and $r_{i_{o,1}}$ and $r_{i_{o,2}}$ for $r_{s_{s,1}}$ and $r_{s_{s,2}}$, respectively. Similarly, the argument is analogous for $r_{i,2}$, by replacing $r_{s_{s,1}}$ and $r_{s_{s,2}}$ with $r_{i_{o,1}}$ and $r_{i_{o,2}}$, respectively, and $h_{s_{s,1}}$ and $h_{s_{s,2}}$ with $h_{i_{o,1}}$ and $h_{i_{o,2}}$, respectively.

**Lemma 3.3.** For all $i$, $r_{i,3}$ and $r_{i,4}$ are matched to some hospital in $M'$. Moreover, $r_{i,3}$ and $r_{i,4}$ are matched to the same hospital in $M'$.

**Proof.** If $r_{i,3}$ is not matched in $M'$, he clearly forms a blocking pair with $h_{i,1}$, who has $r_{i,3}$ first on his preference list. Similarly, if $r_{i,4}$ is not matched, he blocks with $h_{i,1}$.

For the second claim, suppose $(r_{i,3}, h_{i,1})$ and $(r_{i,4}, h_{i,2})$ are in $M'$. Then $r_{i,1}$ cannot be matched to $h_{i,1}$ and $r_{i,2}$ cannot be matched to $h_{i,2}$ in $M'$, for otherwise the capacities of $h_{i,1}$ and $h_{i,2}$ would be exceeded. However, this implies $(r_{i,1}, h_{i,1})$ and $(r_{i,2}, h_{i,2})$ form
blocking pairs for $M'$. On the other hand, if $(r_{i,3}, h_{i,1})$ and $(r_{i,4}, h_{i,1})$ are in $M'$, then $(r_{i,3}, h_{i,1})$ form a blocking pair in $M'$, for $h_{i,1}$ has enough spare capacity to admit $r_{i,3}$.

**Lemma 3.4.** For all $i$, exactly one of $\{r_{i,1}, r_{i,2}\}$ is matched to his first choice, and the other is matched to his second choice in $M'$.

**Proof.** By Lemma 3.3, it is clear that $r_{i,1}$ and $r_{i,2}$ cannot both be matched to their first choice in $M'$, for this would result in $r_{i,3}$ and $r_{i,4}$ being unassigned in $M'$, a contradiction.

On the other hand, if $r_{i,1}$ and $r_{i,2}$ are both matched to their second choice, then if $r_{i,3}$ and $r_{i,4}$ are both matched to $h_{i,1}$, then $r_{i,2}$ forms a blocking pair with $h_{i,2}$. If instead $r_{i,3}$ and $r_{i,4}$ are both matched to $h_{i,2}$, then $r_{i,1}$ forms a blocking pair with $h_{i,1}$.

Finally, by Lemma 3.2, $r_{i,1}$ and $r_{i,2}$ cannot be unmatched or matched to the last hospitals on their preference lists, so exactly one of $\{r_{i,1}, r_{i,2}\}$ is matched to his first choice in $M'$, and the other to his second.

Lemmas 3.2-3.4 lead us to the following corollary.

**Corollary 3.5.** Every resident is matched in any stable matching $M'$ for $I'$.

**Proof.** The only residents not yet shown to be matched in $M'$ are those residents $r_{i,k}$ for $\delta \in \{\alpha, \beta\}$ and $k \in \{1, 2\}$ created from a man $m_i$ with a preference list consisting of a tie of size 2 in $I$, and the residents $r_{\delta, k}$ for $k \in \{1, 2\}$ created from a man $m_\delta$ with a strictly ordered preference list in $I$. Each of these residents must be matched in $M'$, for otherwise they would form a blocking pair with the last hospital on their preference list.

We are now in a position to prove the first direction of the reduction.

**Lemma 3.6.** If the derived HRS instance $I'$ admits a stable matching $M'$, then the given instance $I$ of (3,3)-COM-SMTI admits a complete stable matching $M$.

**Proof.** Given a stable matching $M'$ for $I'$, we describe how to construct a complete stable matching $M$ in $I$ as follows. Consider the residents $r_{i,k}$ for $k \in \{1, 2, 3, 4\}$ that were created in correspondence to a man $m_i$ in $I$ with a preference list consisting of $(w_k, w_l)$, a tie of size 2, where $k < l$. By Lemma 3.4, either $r_{i,1}$ is matched to $h_k$ or $r_{i,2}$ is matched to $h_l$ in $M'$, and, since the capacity of every hospital in $I'$ is either 2 or 1, no other resident is assigned to $h_k$ if $r_{i,1}$ is, and similarly for $r_{i,2}$ and $h_l$. Hence, we construct $M$ by placing $(m_i, w_k)$ into $M$ if and only if $(r_{i,3}, h_k) \in M'$, and $(m_i, w_l)$ into $M$ if and only if $(r_{i,2}, h_l) \in M'$. Again, Lemma 3.4 ensures we always place exactly one such pair into $M$. To complete the construction of $M$, for each resident $r_i$ corresponding to a man $m_i$ with a strictly ordered preference list, place $(m_i, w_j)$ into $M$ if and only if $(r_i, h_j) \in M'$. Corollary 3.5 ensures every man in $I$ is assigned to some woman in $M$; in what follows we will show that $M$ is indeed a matching, and is also stable.

We have already argued by Lemma 3.4 that no two men with ties on their preference lists are matched to the same woman in $M$. For any resident $r_i$ corresponding to a man $m_i$ in $I$ with a strictly ordered preference list, $r_i$ must have size 2, and is matched to a hospital $h_j$, which is not his last choice by Lemma 3.2, and which therefore corresponds to woman $w_j$ in $I$. Hospital $h_j$ has capacity 2, and so is matched to only $r_i$ in $M'$. This means exactly one man is matched to $w_j$ in $M$. Therefore, $M$ is a matching.

We will show $M'$ is stable by demonstrating that no man can be part of a blocking pair relative to $M$. For any man $m_i$ with a preference list consisting of a tie of size two, this must true, for they are indifferent between the only two women on their preference list. Suppose instead $m_i$ has a strictly ordered preference list. Consider any woman $w_l$ whom $m_i$ prefers in $M$. Then, in $M'$, resident $r_i$ must also have preferred hospital $h_l$. 

9
By Lemma 3.2 and the construction of the hospitals of $I'$, $h_l$’s preference list contains residents of size 2 only. Since $M'$ is stable, $h_l$ is assigned a resident it strictly prefers over $r_i$, and hence $w_l$ is assigned a man she strictly prefers over $m_i$ in $M$. It follows that $M$ is a complete stable matching in $I$. 

We now prove the reduction is correct in the other direction.

**Lemma 3.7.** If the given instance $I$ of $(3,3)$-COM-SMTI admits a complete stable matching $M$, then the derived HRS instance $I'$ admits a stable matching $M'$.

**Proof.** Given a complete stable matching $M$ for $I$, we describe how to construct a complete stable matching $M'$ in $I$. For each man $m_i$ with a strictly ordered preference list, place $(r_i, h_j)$ into $M'$ if and only if $(m_i, w_j) \in M$. For each man $m_i$ in $M$ with a tie of size 2 consisting of, say, $(w_k, w_l)$, where $k < l$, construct $M'$ by the following two rules:

1. If $(m_i, w_k) \in M$, place $(r_{i,1}, h_k), (r_{i,2}, h_k), (r_{i,3}, h_{i,1}), (r_{i,4}, h_{i,1})$ into $M'$, and assign all residents $r_{i,k} \forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ with their first choice.

2. If $(m_i, w_l) \in M$, place $(r_{i,1}, h_{i,1}), (r_{i,2}, h_l), (r_{i,3}, h_{i,2}), (r_{i,4}, h_{i,2})$ into $M'$, and assign all residents $r_{i,k} \forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ with their first choice.

It is easy to verify that the capacities of each hospital are not exceeded in $M'$, and that $M'$ is a matching. We claim that $M'$ is also stable.

For, suppose residents $r_{i,t}$ for $t \in \{1, 2, 3, 4\}$ are matched by Rule 1 above. Immediately we may notice that $r_{i,2}$ and $r_{i,3}$ are matched with their first choices, and hence cannot form a blocking pair with any hospital in $I'$. Resident $r_{i,1}$ prefers only hospital $h_{i,1}$ to his assignment in $M'$, but does not form a blocking pair with it because $r_{i,3}$ and $r_{i,4}$ are matched to $h_{i,1}$. The remaining resident, $r_{i,4}$ prefers only $h_{i,2}$, who is matched with $r_{i,2}$, and hence does not form a blocking pair with $r_{i,4}$. All residents $r_{i,k} \forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ are matched with their first choice and cannot form a blocking pair with any hospital.

Suppose residents $r_{i,t}$ for $t \in \{1, 2, 3, 4\}$ are matched by Rule 2 above. In a symmetric argument to the previous rule, $r_{i,1}$ and $r_{i,4}$ are matched with their first choices, and cannot be part of a blocking pair. Resident $r_{i,2}$ prefers only hospital $h_{i,2}$ to his assignment in $M'$, but does not form a blocking pair with it because $r_{i,3}$ and $r_{i,4}$ are matched to $h_{i,2}$. The remaining resident, $r_{i,3}$ prefers only $h_{i,1}$, who is matched with $r_{i,1}$, and hence cannot form a blocking pair with $r_{i,3}$. Again, the residents $r_{i,k} \forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ are matched with their first choice and cannot form a blocking pair with any hospital.

In the final case, a resident $r_i$ corresponding to a man $m_i$ with a strictly ordered preference list in $I'$ cannot block for the same reasons that $m_i$ did not block in $M$. If $m_i$ preferred a woman $w_j$ in $M$, then $r_i$ must also prefer $h_j$ in $M'$. However, $h_j$ must be matched to a resident that precedes $r_i$ on its preference list, since $w_j$ is matched to a man preceding $m_i$ on her preference list. The capacity of $h_j$ is 2, and the size of $r_i$ is also 2, so that $r_i$ cannot be “added” to $h_j$. Therefore, $M'$ is a stable matching for $I'$.

Lemmas 3.6 and 3.7 immediately imply the following theorem.

**Theorem 3.8.** The problem of determining whether an HRS instance admits a stable matching is NP-complete, even if the size of each resident and the capacity of each hospital is at most 2, and the lengths of the residents’ and hospitals’ preference lists are at most 3 (these conditions holding simultaneously).

The following corollary follows immediately by Theorem 3.8 and Lemma 2.1.
Corollary 3.9. The problem of determining whether an HRCC instance admits a stable matching is NP-complete, even if the individual preference list of each resident and the joint preference list of each couple has at most 3 entries, and the capacity of each hospital is at most 2 (these conditions holding simultaneously).

4 HRCC under classical (Gale-Shapley) stability

We begin this section by defining the variant of HRCC in which stability is defined with respect to classical (Gale-Shapley) stability. We provide a linear time algorithm for this problem, without any assumptions about the lengths of the preference lists or capacities of the hospitals. The problem is defined in the same manner as HRCC, the difference, however, lies in the definition of a blocking pair. So, as before, each hospital $h_j \in H$ provides a preference list of acceptable residents, denoted $L_{h_j}$, and each resident $r_i \in R$ (whether they are a member of a couple or not) submits an individual preference list $L_{r_i}$ of acceptable hospitals. Each couple $c_k$ then constructs a joint preference list $L_{c_k}$ that is consistent as defined in Section 2. A blocking pair for a matching, then, is defined to be a (resident, hospital) pair $(r_i, h_j)$ such that (i) according to $L_{r_i}$, $r_i$ prefers $h_j$ to $M(r_i)$ and (ii) either $h_j$ is undersubscribed, or according to $L_{h_j}$, $h_j$ prefers $r_i$ to at least one member of $M(h_j)$. Notice the difference in the stability definition to that defined in Section 2 for HRC is that blocking pairs are defined with respect to the individual preference lists, rather than the couples’ joint preference lists. We partition the set of preference lists $L$ of an instance of HRCC into three sets $L = L^C \cup L^R \cup L^H$ where $L^R$ is the set of individual preference lists of the residents, $L^H$ is the set of hospitals’ preference lists, and $L^C$ is the set of joint lists created by the couples. The goal in this setting is to find a matching satisfying the following two criteria:

1. The matching contains no blocking pairs relative to $L^R$ and $L^H$ under the classical definition of Gale-Shapley stability as defined above.

2. Each couple $c_k = (r_i, r_j)$ is assigned to a pair of hospitals $(h_p, h_q) \in L_{c_k}$ or both $r_i$ and $r_j$ are unassigned.

The instance induced by the preference lists $L^R$ and $L^H$ is a classical Hospitals / Residents instance, so finding a matching satisfying (1) above simply involves using the extended Gale-Shapley algorithm to compute a stable matching $M$. Of course, $M$ may not satisfy (2), in which case we need to determine if there is a different stable matching which does. Efficient algorithms are known for enumerating the set of all stable matchings [8], however, there may be exponentially many of them [15]. Thus we need a direct approach to determine if a matching satisfying (1) and (2) exists. Henceforth, let such a stable matching be called a feasible stable matching.

Given an instance $I$ of HRCC, let $M$ denote the set of all stable matchings under classical stability with respect to $L^R$ and $L^H$. We shall obtain a polynomial time algorithm for determining the existence of a feasible stable matching by exploiting the known results about the rich structure of $M$. We begin with the following so-called rural hospitals theorem [18, 7] (see also [9, Section 1.6.5]).

Theorem 4.1. For any given hospitals/residents instance, (i) each hospital is assigned the same number of residents in all stable matchings; (ii) exactly the same set of residents are unassigned in all stable matchings; (iii) any hospital which is undersubscribed in one stable matching is matched with precisely the same set of residents in all stable matchings.
Since our goal in this section is to develop an algorithm to match couples together, part (ii) of the previous theorem implies that if for some instance of the problem we have a stable matching in which it is the case that one member of a couple is assigned and the other is unassigned, no feasible stable matching exists, for there is no stable matching in which they are either both assigned or both unassigned. By the same token, we cannot in general trivially obtain a feasible stable matching by selecting every resident in a couple to simply be unassigned. We continue with the following known relation which induces a partial order on $M$ [9].

**Definition 4.2.** Let $M$ and $M'$ be stable matchings for an HR instance. We say that $M$ dominates $M'$ (denoted $M \succeq M'$) if, for each assigned resident $r$, $M(r) = M'(r)$, or $r$ strictly prefers $M(r)$ to $M'(r)$. Intuitively, $M$ dominates $M'$ if each resident is at least as happy in $M$ as in $M'$ (the case that $M \succeq M'$ and $M \neq M'$ is denoted by $M \succ M'$).

Notice in fact that a stable matching dominates itself. While we do not necessarily need the result here, it is interesting to note that the set of stable matchings along with $\succeq$ forms a distributive lattice. When the extended Gale-Shapley algorithm is run, if the sequence of proposals come from the residents, the resulting matching is the resident-optimal stable matching, denoted $M_R$, in which each matched resident is assigned to the best hospital he can ever be assigned to in any stable matching, whilst each unmatched resident is unmatched in every stable matching [9, Theorem 1.6.2]. If instead the sequence of proposals come from the hospitals, the resulting stable matching is the hospital-optimal stable matching, denoted $M_H$, in which each undersubscribed hospital $h_j$ is assigned the residents in $M_H(h_j)$ in every stable matching, and each full hospital $h_j$ is assigned its best $c_j$ partners in any stable matching [9, Theorem 1.6.1]. $M_R$ and $M_H$ are the maximum and minimum elements, respectively of the lattice of stable matchings [9, Section 1.6.5]. It is this underlying structure of $M$ that will allow us to develop the efficient algorithm presented in this section. We continue with the following proposition [9, Section 1.6.5].

**Proposition 4.3.** The resident-optimal stable matching $M_R$ dominates all stable matchings in $M$.

### 4.1 Breakmarriage

The algorithm we develop will use as a subroutine a generalised version of an algorithm known as Algorithm Breakmarriage, first defined by McVitie and Wilson [16] and used again by Gusfield [8]. Our modified version of Algorithm Breakmarriage takes as input any stable matching $M \neq M_H$, and takes as a second parameter any resident $r$ who is matched in $M$ such that $M(r) \neq M_H(r)$, and always outputs a new stable matching dominated by $M$. A description of this algorithm is as follows:

**Breakmarriage**$(M, r)$

Given the stable matching $M$ and a matched resident $r$ as input, let $R' \subseteq R$ denote the set of residents $r'$ matched to $M(r) = h$ in $M$ such that $r' = r$ or $r'$ succeeds $r$ on the preference list of $h$. Restart the extended Gale-Shapley algorithm by unassigning all pairs $(r', h)$ for all $r' \in R'$. All residents in $R'$ are now free and are pushed onto a stack $S$ in arbitrary order. Hospital $h$ is defined to be “semi-free” in that it only accepts new proposals from residents it strictly prefers to $r$. Algorithm Breakmarriage iteratively pops a resident $r''$ from $S$ with $r''$ proposing to the first hospital following $M(r'')$ on his preference list, and this initiates a sequence of proposals, rejections, and acceptances as given by the resident-oriented Gale-Shapley algorithm [9, Section 1.6.3] in which free residents that have not become rejected by every hospital on their preference list become
pushed onto $S$. Algorithm Breakmarriage terminates when $S$ becomes empty. The current set of assignments $M'$ is then output.

The following facts hold about Algorithm Breakmarriage.

**Proposition 4.4.** Suppose $M$ and $M'$ are stable matchings such that $M \succeq M'$, and that resident $r$ is matched in $M$ and satisfies $M(r) \neq M'(r)$. Then, in the resulting execution of Algorithm Breakmarriage($M, r$), no resident $r' \in R$ ever proposes to a hospital succeeding $M'(r')$ on his preference list (in the case that $M(r') = M'(r')$, this implies that $r'$ remains matched to $M(r')$ in the execution of Breakmarriage($M, r$)).

**Proof.** Let $h = M(r)$, and let $R' \subseteq R$ denote the set of residents $r' \in M(h)$ such that $r' = r$ or $r'$ succeeds $r$ on the preference list of $h$. Since $r \not\in M'(h)$ and $M \succeq M'$, it follows that $r$ prefers $h$ to $M'(r)$. Hence $h$ is full in $M'$ and prefers each of its assignees in $M'$ to $r$. It follows that $h$ prefers each of its assignees in $M'$ to each member of $R'$.

Now suppose that a resident $r'$ who is matched in $M$ is the first resident in the execution of Algorithm Breakmarriage($M, r$) to be rejected by the hospital he is assigned to in $M'$. Let $h' = M'(r')$. Clearly if $h' = h$ and $r' \in R'$ then $M(r') = M'(r')$, which is impossible by the first paragraph. Hence $r'$ was rejected by $h'$ during the phase of Algorithm Breakmarriage that corresponds to the restart of the resident-oriented Gale-Shapley algorithm. Let $M_A$ be the matching at the point during the execution of the algorithm when $h'$ rejected $r'$. Then $h'$ is full in $M_A$ and prefers each of its assignees in $M_A(h')$ to $r'$. Since $r' \in M'(h') \setminus M_A(h')$ and $h'$ is full in $M_A$, it follows that there exists some $r_w \in M_A(h') \setminus M'(h')$.

If $r_w$ is matched in $M'$, then $r_w$ cannot yet have proposed to $M'(r_w)$ (as $h' \neq M'(r_w)$, and hence this would contradict the fact that $r'$ is the first resident to be rejected by the hospital that he is assigned to in $M'$). Hence either $r_w$ is unmatched in $M'$ and finds $h'$ acceptable, or $r_w$ prefers $h'$ to $M'(r_w)$. But this implies that $(r_w, h')$ form a blocking pair for $M'$, as $h'$ prefers $r_w$ to $r'$. Therefore, $r'$ is not rejected by $h'$ in the call to Algorithm Breakmarriage($M, r$).

**Corollary 4.5.** Let $M \neq M_H$ be a stable matching and $r$ an arbitrary resident with $M(r) \neq M_H(r)$. Then, no resident $r' \in R$ is ever rejected by $M_H(r')$ in a call to Algorithm Breakmarriage($M, r$).

**Proposition 4.6.** When Algorithm Breakmarriage($M, r$) terminates, the set of assignments $M'$ output by the algorithm is a stable matching.

**Proof.** We first observe that $M(r)$ is full in $M$, by Theorem 4.1, since $M(r) \neq M_H(r)$. We proceed by showing that every hospital that is full in $M$ is also full in $M'$. Throughout the execution of Algorithm Breakmarriage($M, r$), no hospital that was full in $M$ can become undersubscribed except for $M(r)$, as no other hospital rejects a resident without gaining a better one. Therefore, the set of hospitals that are undersubscribed at some point in the execution of the algorithm are those that are undersubscribed in every stable matching (by Theorem 4.1) and $M(r)$. Suppose that a resident $r'$ were to propose to a hospital $h'$, such that $h'$ is undersubscribed in $M$, during the algorithm’s execution. By Theorem 4.1, $h'$ is undersubscribed in $M_H$, and also $(r', h') \notin M_H$, since $(r', h') \notin M$. If $r'$ is unmatched in $M_H$ then $(r', h')$ blocks $M_H$, a contradiction. Hence $r'$ is matched in $M_H$. As $h' \neq M_H(r')$, by Corollary 4.5, $h'$ precedes $M_H(r')$ on $r'$’s preference list, implying that $(r', h')$ blocks $M_H$, a contradiction. Hence, no resident may propose to a hospital that is undersubscribed in $M$ at any point in the execution of the algorithm, implying that proposals are only made to those hospitals that are full in $M$. It follows by a simple counting argument that $M(r)$ is full in $M'$.
To see that \( M' \) is stable, we know that any resident other than those in \( R' \) has not been rejected by a hospital without proposing to it, and hospitals only improve throughout the execution of the algorithm, so that these residents cannot be a part of a blocking pair. For any resident \( r' \in R', \) \( r' \) also is not rejected by a hospital without proposing to it, with the exception of hospital \( M(r'). \) By the above discussion, \( M(r') \) is full in \( M' \), and by the description of the algorithm, \( M(r') \) only accepts proposals from residents it prefers to \( r' \), hence \( r' \) cannot be a part of a blocking pair either.

**Proposition 4.7.** Let \( M \) and \( M' \) be distinct stable matchings, and suppose that \( M \) dominates \( M' \). Then, if \( r \) is a resident with \( M(r) \neq M'(r) \), Algorithm Breakmarriage either returns \( M' \) or a stable matching \( M'' \) that dominates \( M' \) (i.e., \( M \succeq M'' \succeq M' \)).

**Proof.** This is an immediate consequence of Propositions 4.4 and 4.6.

**Proposition 4.8.** Any stable matching \( M \) can be obtained by a series of calls to Algorithm Breakmarriage from the resident-optimal stable matching \( M_R \) in \( O(L) \) time, where \( L \) is the sum of the lengths of the preference lists.

**Proof.** The fact that an arbitrary stable matching \( M \) can be obtained from \( M_R \) by a series of calls to Algorithm Breakmarriage is a straightforward consequence of Proposition 4.7. A total of \( O(L) \) time is spent, as any arbitrary series of calls to the algorithm constitutes at most one left to right traversal of each resident’s preference list, and similar time is spent traversing the hospitals’ preference lists.

So, in light of Proposition 4.8, we can see that an approach to computing a feasible stable matching (if one exists) can be achieved by first finding the resident-optimal stable matching \( M_R \), and making a suitable selection of calls to Algorithm Breakmarriage. In the next subsection, we will show that because the preference lists in \( L_{c_k} \) are consistent, we can always compute an appropriate sequence of calls to Algorithm Breakmarriage in linear time.

We also note that repeated calls to Algorithm Breakmarriage ultimately yield the hospital-optimal stable matching, in which every resident is, of course, assigned to \( M_H(r) \). Thus this is the only stable matching Algorithm Breakmarriage cannot take as input.

### 4.2 The algorithm

Before presenting the main algorithm of this section, we require some preliminary lemmas and definitions. Let \( M \) be a stable matching. Recall Theorem 4.1, which states that precisely the same set of residents are matched in every stable matching. With this in mind, we define a **matched couple** \( c_k = (r_i, r_j) \) to be a couple such that \( r_i \) and \( r_j \) are matched in \( M \) (and hence in every stable matching). Similarly, we define an **unmatched couple** \( c_k = (r_i, r_j) \) to be a couple such that one or both of \( r_i \) and \( r_j \) are unmatched in \( M \) (and hence in every stable matching). Let \( c_k = (r_i, r_j) \) be a matched couple. We define the **next acceptable pair** on \( L_{c_k} \) (denoted \( next_M(c_k) \)) to be the first pair of hospitals \( (h_p, h_q) \) on \( L_{c_k} \) such that \( h_p \) succeeds or is equal to \( M(r_i) \) on \( L_{r_i} \) and \( h_q \) succeeds or is equal to \( M(r_j) \) on \( L_{r_j} \). If no such pair exists, we say \( next_M(c_k) \) = \( \emptyset \) with slight abuse of notation.

**Example** To illustrate the notion of the next acceptable pair for a couple, we refer the reader to Figure 3. This shows an HRCC instance with 8 residents \( r_1, r_2, \ldots, r_8 \) and 5 hospitals \( h_1, h_2, \ldots, h_5 \). There are a total of three couples, namely \((r_2, r_3), (r_4, r_5), \) and \((r_7, r_8). \) A stable (but not feasible) matching \( M \) for this instance is denoted by underlining. In \( M, next_M(r_2, r_3) = (h_3, h_2), next_M(r_4, r_5) = (h_4, h_4), \) and \( next_M(r_7, r_8) = (h_2, h_2). \)
Lemma 4.9. Let $M$ be a non-feasible stable matching that dominates a feasible stable matching $M_f$. Let $c_k = (r_i, r_j)$ be any matched couple who are not matched to a pair of hospitals on $L_{c_k}$. Then, in $M_f$, $(r_i, r_j)$ is either assigned to $next_M(c_k)$ or a pair of hospitals succeeding $next_M(c_k)$ on $L_{c_k}$.

Proof. Since $M$ dominates $M_f$, each resident $r$ either has $M(r) = M_f(r)$ or $M_f(r)$ succeeds $M(r)$ on $L$, by Definition 4.2. So, for a couple $c_k = (r_i, r_j)$, their partners in $M_f$ are either their current hospitals or hospitals that appear further down their individual preference lists. It follows then, that $next_M(c_k)$ is the first pair of hospitals on $L_{c_k}$ that $c_k$ could be assigned to in $M_f$. So, in $M_f$, $c_k$ is either assigned to $next_M(c_k)$ or to a pair of hospitals that succeeds $next_M(c_k)$ on $L_{c_k}$. \hfill \Box

Lemma 4.10. Let $M$ be a non-feasible stable matching that dominates a feasible stable matching $M_f$. Let $c_k = (r_i, r_j)$ be any matched couple who are not matched to a pair of hospitals on $L_{c_k}$. Let $(h_p, h_q) = next_M(c_k)$. Then,

1. Either $M(r_i) \neq h_p$ or $M(r_j) \neq h_q$ (or both). Let $r^*$ denote whichever of $r_i$ or $r_j$ satisfy (1).

2. The stable matching obtained by calling Algorithm Breakmarriage with $M$ and $r^*$ dominates $M_f$.

Proof. For the first claim, $r_i$ and $r_j$ cannot both be assigned to $h_p$ and $h_q$, respectively in $M_f$, for $M_f$ is feasible, and this pair does not appear on $L_{c_k}$, by the assumption of the lemma. Let $r^*$ denote whichever of $r_i$ and $r_j$ have $M(r^*) \neq M_f(r^*)$, choosing arbitrarily if both do.

For the second claim, since $M$ is not feasible and $M$ dominates $M_f$, the members of $c_k$ must be assigned to $next_M(c_k)$ or to a pair of hospitals succeeding $next_M(c_k)$ by Lemma 4.9. By the nature of the construction of the joint preference list $L_{c_k}$, and by the fact that $M$ dominates $M_f$, this implies that $r^*$ is assigned to a hospital succeeding $M(r^*)$ on $L_{c_k}$ in $M_f$. Hence by Proposition 4.7, calling Algorithm Breakmarriage on the current matching and $r^*$ yields a stable matching that dominates $M_f$. \hfill \Box

Figure 3: An HRCC instance with a stable but not feasible matching
We are now ready to describe the algorithm for finding a feasible stable matching or reporting “none exists”. The algorithm begins by computing the resident-optimal stable matching \( M_R \) and the hospital-optimal stable matching \( M_H \). By Proposition 4.3, \( M_R \) dominates all stable matchings in \( M \) – hence it dominates every feasible stable matching (if any exist) as well. If \( M_R \) is itself feasible, the algorithm returns \( M_R \). Otherwise, if for any couple \((r_i, r_j)\) it is the case that \( r_i \) is assigned and \( r_j \) is unassigned, the algorithm halts, correctly reporting that no feasible stable matching exists by Theorem 4.1.

Only if no such couple exists do we enter the while loop which maintains the loop condition that the current matching \( M \) is not feasible – hence there is some couple \( c_k = (r_i, r_j) \) who are not assigned to a hospital on \( L_{c_k} \). If \( \text{next}_M(c_k) = \emptyset \), the algorithm outputs “No feasible stable matching exists”. Otherwise the algorithm identifies a resident \( r^* \in \{r_i, r_j\} \) such that \( M(r^*) \) is not equal to \( r^* \)’s partner in \( \text{next}_M(c_k) \). If \( r^* \) has the same partner in \( M \) and in \( M_H \), the algorithm outputs “No feasible stable matching exists”. Otherwise, we call Algorithm Breakmarriage with \( M \) and \( r^* \). The loop is exited only when the algorithm outputs “No feasible stable matching exists” or when the current matching \( M \) is feasible. The pseudocode for the algorithm is presented in Figure 4 as Algorithm HRCC.

### 4.2.1 Correctness

If no feasible stable matching exists, Algorithm HRCC will clearly correctly output “no feasible stable matching exists” in one of three places. If, before entering the while loop, it is the case that there is a couple with one member assigned and the other unassigned, the algorithm correctly halts. Otherwise, the algorithm enters the while loop, and since no feasible stable matching exists, the algorithm continues to make calls to Algorithm Breakmarriage. This process must eventually halt, either when \( \text{next}_M(c_k) = \emptyset \) for some couple \( c_k \), or when the successive calls to Algorithm Breakmarriage eventually yield \( M_H \), at either point Algorithm HRCC will correctly output in the negative.

So, instead, let us suppose a feasible stable matching \( M_f \) does exist. We claim that Algorithm HRCC maintains the invariant that at each iteration of the while loop the current matching \( M \) dominates \( M_f \). The claim is clearly true when \( M = M_R \), by Proposition 4.3, so let us assume the invariant is true at the end of the \( i \)th iteration. Let \( M_i \) denote the stable matching at the end of iteration \( i \) of the while loop and suppose that \( M_i \) is not feasible. Since \( M_i \) is not feasible, there is some assigned couple \( c_k = (r_s, r_t) \) that is not assigned a pair from \( L_{c_k} \). By Lemma 4.10, there is at least one resident \( r^* \in \{r_s, r_t\} \) that is not assigned to a hospital in the ordered pair \( \text{next}_M(c_k) \), and, further, calling Algorithm Breakmarriage on the current matching \( M_i \) and \( r^* \) yields a stable matching that dominates \( M_i \) by Proposition 4.7, since \( M_i \) dominates \( M_f \). Thus the matching \( M_{i+1} \) obtained by this process dominates \( M_f \), and the claim follows.

Thus, at each iteration of the while loop of Algorithm HRCC, the current matching dominates \( M_f \). Hence, the algorithm eventually terminates having encountered \( M_f \) or a different feasible stable matching that dominates \( M_f \). Let \( M_f^* \) denote the feasible stable matching that is returned by the algorithm. Since \( M_f \) is an arbitrary feasible stable matching, we have argued that \( M_f^* \) dominates every feasible stable matching. Hence, \( M_f^* \) is resident-optimal amongst the set of feasible stable matchings.

We summarise this section with the following theorem.

**Theorem 4.11.** Algorithm HRCC finds the resident-optimal feasible stable matching \( M_f^* \) if it exists or reports “none exists” in \( O(L) \) time, where \( L \) is the sum of the lengths of the preference lists of the input.
Compute $M_R$ and $M_H$;
$M \leftarrow M_R$;
for each couple $c \in C$
    if (one member of $c$ is assigned in $M$ and the other is unassigned in $M$)
        report “no feasible stable matching exists”;
        HALT;
while (some couple $c_k = (r_i, r_j)$ is not assigned a pair from $L_{c_k}$)
    if (some matched couple $c_k$ has $next_M(c_k) = \emptyset$)
        report “no feasible stable matching exists”;
        HALT;
    $r^* \leftarrow$ a resident in $c_k$ with a different partner in $M$ and $next_M(c_k)$
    if ($M(r^*) = M_H(r^*)$)
        report “no feasible stable matching exists”;
        HALT;
    else
        $M \leftarrow \text{Breakmarriage}(M, r^*)$;
return($M$);

Figure 4: Algorithm HRCC

Proof. We have shown that the algorithm finds the resident-optimal feasible stable matching if it exists, or reports “none exists” correctly. To establish the claimed runtime, we observe that the algorithm constitutes essentially a “left to right” sweep of the residents’ preference lists. So, by using appropriate data structures (extending those described in [9, Section 1.2.3] for the Extended Gale-Shapley algorithm for SMI to the HR case), we can implement this algorithm to run in $O(L)$ time.

We end this section with the remark that HRCC under classical stability is a variant of HR that can be solved in polynomial time by a unified approach [3] since this problem exhibits the so-called independence property (see [3] for the definition of this property and further details). For completeness and for consistency with the notation and terminology adopted in the remainder of this paper, we have chosen to present the main result of this section as a standalone algorithm.

5 HRS with hospital preference lists of length $\leq 2$

In light of the NP-completeness result for HRS presented in Section 3, it is natural to ask if, by specialising the problem version, we can identify a “boundary” for which HRS becomes polynomial-time solvable. One option for us to consider is to allow the sizes of the residents to be at most 1, rather than 2. This restriction would, of course, yield an instance of the classical Hospitals / Residents problem, which is polynomial-time solvable. A different option is to further restrict the lengths of the preference lists for the residents and/or the hospitals. We show that by restricting the lengths of the preference list of each hospital to be at most 2, rather than 3, a stable matching always exists, and an extension of the Gale-Shapley algorithm finds a stable matching in polynomial time, even if no restriction is placed on the sizes of the residents, the lengths of the preference lists of the residents, or the capacities of the hospitals. Since NP-completeness for HRS holds even for hospital preference lists of length at most 3, the results of this section indicate such a boundary for HRS. We refer to an instance of HRS in which the lengths of the hospitals’ preference lists are at most 2 and the residents’ lists are unbounded as $(\infty, 2)$-HRS.
assign all residents to be free;
while (some resident \( r_i \) is free and \( r_i \) has a nonempty list)
\[
    h_j \leftarrow \text{first hospital on } r_i \text{'s list};
\]
provisionally assign \( r_i \) to \( h_j \); // in matching \( M \)
if \( (r_i, h_j) \) is \( h_j \text{'s first choice and } h_j \text{'s list is of length 2) \)
\[
    r_k \leftarrow h_j \text{’s second choice};
\]
if \( (O_j^M + s_k > c_j) \)
delete \( (r_k, h_j) \); // from the preference lists and from \( M \)

Figure 5: Algorithm \((\infty, 2)\)-HRS

The procedure for solving \((\infty, 2)\)-HRS is as follows. The algorithm can be seen as a
process of “proposal” operations from the residents to the hospitals. A resident proposes
sequentially to each hospital on his list until he becomes assigned or his list becomes
empty. When a resident \( r_i \) proposes to a hospital \( h_j \), \( r_i \) becomes provisionally assigned to
\( h_j \). If \( r_i \) is that hospital’s first choice, and \( h_j \text{'s preference list has another entry, we let } r_k \)
denote \( h_j \text{’s second choice. If } s_i + s_k > c_j \), the pair \( (r_k, h_j) \) is deleted, meaning that \( r_k \)
is removed from \( h_j \text{’s preference list, and } h_j \) is removed from \( r_k \text{’s preference list. This is the}
only time a (resident,hospital) pair is deleted by the algorithm. The algorithm continues
this process until each resident is either assigned a hospital or has an empty list. The
details of the algorithm are shown in Figure 5.

Let us now establish the correctness and time complexity of the algorithm presented.

**Theorem 5.1.** Algorithm \((\infty, 2)\)-HRS finds a resident-optimal stable matching for an
instance of \((\infty, 2)\)-HRS in \( O(L) \) time, where \( L \) is the sum of the lengths of the preference
lists.

**Proof.** It is clear that the provisional assignments at the termination of Algorithm \((\infty, 2)\)-
HRS is a matching \( M \). We claim that \( M \) is stable. To see this, consider an arbitrary
resident \( r_i \) who is unassigned or prefers a hospital \( h_j \) to his assignment in \( M \). Then, since
\( r_i \) is not assigned to \( h_j \) in \( M \) and prefers \( h_j \) to his current assignment, \( h_j \) must have been
deleted from \( r_i \)’s preference list. But this can only happen if \( r_i \) is \( h_j \text{’s second choice and } h_j \)
was assigned to its first choice at some point in the algorithm and does not have enough
spare capacity to add \( r_i \). But \( h_j \text{’s first choice can never become unassigned from } h_j \) at
any subsequent step of the algorithm – so in fact \( r_i \) cannot block with \( h_j \) in \( M \). Since \( r_i \)
was chosen arbitrarily it follows that no resident is part of a blocking pair in \( M \).

Secondly, we claim that Algorithm \((\infty, 2)\)-HRS never deletes a stable pair (i.e., a
(resident,hospital) pair that belongs to some stable matching). For, suppose that \((r_k, h_j)\)
is the first such pair deleted during an arbitrary execution of the algorithm, and let \( M' \)
be a stable matching containing \((r_k, h_j)\). Then \( r_k \) was deleted because some resident \( r_i \)
preceding \( r_k \) on \( h_j \text{’s preference list became assigned to } h_j \) and \( s_i + s_k > c_j \). Now, since no
stable pair has been deleted prior to this point, in \( M' \), \( r_i \) is either assigned to \( h_j \) or to a
hospital lower than \( h_j \), or is unassigned. Since \( r_i \) and \( r_k \) cannot both be assigned to \( h_j \) in
\( M' \), it follows that \((r_i, h_j)\) blocks \( M' \), a contradiction.

Thus we have shown that \( M \) is stable and that each resident is assigned to their optimal
partner in \( M \). Lastly, with the aid of data structures similar to those used to achieve an
\( O(L) \) implementation of the Gale-Shapley algorithm for HR [9], we can achieve a similar
complexity for Algorithm \((\infty, 2)\)-HRS. \qed
6 Concluding remarks

Our stability definition for HRS allows a resident $r_i$ to displace a group of inferior residents of a given total size, so long as this frees up enough space for $r_i$. This could, of course, include a situation whereby a resident of size 10 is displaced in order to make way for a resident of size 1, for example. Our definition assumes that the quality of the assignees takes precedence over the size. However it may be the case that a hospital’s primary concern is to ensure that its occupancy is as high as possible. Thus it would not participate in a blocking pair if its occupancy were to reduce as a result of rejecting the inferior residents and taking on the new resident. This gives rise to an alternative stability definition which is obtained from the one given for HRS in Section 2 by modifying Condition 2 as follows:

2. $O_j^M + s_i \leq c_j$, or $h_j$ prefers $r_i$ to residents $r_{k_1}, \ldots, r_{k_t} \in M(h_j)$ such that

$$s_i \geq \sum_{p=1}^{t} s_{k_p} \quad \text{and} \quad O_j^M + s_i - \sum_{p=1}^{t} s_{k_p} \leq c_j.$$ 

It remains open to investigate the algorithmic complexity of the problem of finding a matching that satisfies this new version of stability, for a given HRS instance.

Acknowledgement

We would like to thank Rob Irving, Ildikó Schlotter, and two anonymous referees for helpful comments on earlier versions of this paper.

References


Appendix: NP-completeness of (3,3)-COM-SMTI

Proof of Theorem 3.1. We reduce from a restricted version of SAT. Let (2,2)-E3-SAT denote the problem of deciding, given a Boolean formula $B$ in CNF in which each clause contains exactly 3 literals and, for each variable $v_i$, each of literals $v_i$ and $\bar{v}_i$ appears exactly twice in $B$, whether $B$ is satisfiable. Berman et al. [1] showed that (2,2)-E3-SAT is NP-complete.
Hence let $B$ be an instance of (2,2)-E3-SAT. Let $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ be the set of variables and clauses respectively in $B$. Then for each $v_i \in V$, each of literals $v_i$ and $\overline{v}_i$ appears exactly twice in $B$. (Hence $m = \frac{4n}{3}$.) Also $|c_j| = 3$ for each $c_j \in C$. We form an instance $I$ of (3,3)-COM-SMTI as follows. The set of men in $I$ is $X = P \cup U \cup Q$, where $X = \bigcup_{i=0}^{n-1} X_i$, $X_i = \{x_{4i+r} : 0 \leq r \leq 3\} \cup \{y_{4i+r} : 0 \leq r \leq 3\}$, $P = \bigcup_{j=0}^{m-1} P_j$, $P_j = \{p^i_j, p^2_j, p^3_j\}$ (1 \leq j \leq m), $U = \bigcup_{j=i}^{m} U_j$, $U_j = \{u^1_j, u^2_j, u^3_j\}$ (1 \leq j \leq m), and $Q = \{q_j : 1 \leq j \leq m\}$. The set of women in $I$ is $Y = \bigcup_{i=0}^{n-1} Y_i$, $Y_i = \{y_{4i+r} : 0 \leq r \leq 3\}$, $\hat{C} = \{c_j : c_j \in C \land 1 \leq s \leq 3\}$, $W = \{w_j : 1 \leq j \leq m\}$, $Z = \bigcup_{j=1}^{m} Z_j$ and $Z_j = \{z^1_j, z^2_j, z^3_j\}$ (1 \leq j \leq m).

The preference lists of the men and women in $I$ are shown in Figure 6. In a given preference list, entries within round brackets are tied. In the preference list of an agent $x_{4i+r} \in X$ (0 \leq i \leq n-1 and r \in \{0, 1, 2\}) holds the symbol $c(x_{4i+r})$ denotes the woman $c^s_j \in C'$ such that the ($r+1$)th occurrence of literal $v_i$ appears at position $s$ of $c_j$. Similarly if $r \in \{2, 3\}$ then the symbol $c(x_{4i+r})$ denotes the woman $c^s_j \in C'$ such that the ($r-1$)th occurrence of literal $\overline{v}_i$ appears at position $s$ of $c_j$. Also in the preference list of an agent $c^s_j \in C'$, if literal $v_i$ appears at position $s$ of clause $c_j \in C$, the symbol $x(c^s_j)$ denotes the man $x_{4i+r-1} = 1, 2$ according as this is the first or second occurrence of literal $v_i$ in $B$. Otherwise if literal $\overline{v}_i$ appears at position $s$ of clause $c_j \in C$. The symbol $x(c^s_j)$ denotes the man $x_{4i+r-1}$ where $r = 1, 2$ according as this is the first or second occurrence of literal $\overline{v}_i$ in $B$. Clearly each preference list is of length at most 3, the men’s lists are strictly ordered, and each woman’s list is either strictly ordered or is a tie of length 2.

For each $i$ (0 \leq i \leq n-1), let $T_i = \{(x_{4i+r}, y_{4i+r}) : 0 \leq r \leq 3\}$ and $F_i = \{(x_{4i+r}, y_{4i+r+1}) : 0 \leq r \leq 3\}$, where addition is taken modulo 4.

We claim that $B$ is satisfiable if and only if $I$ admits a complete stable matching.

For, let $f$ be a satisfying truth assignment of $B$. Define a complete matching $M$ in $I$ as follows. For each variable $v_i \in V$, if $v_i$ is true under $f$, add the pairs in $T_i$ to $M$, otherwise add the pairs in $F_i$ to $M$. Now let $c_j \in C$. As $c_j$ contains a literal that is true under $f$, let $s \in \{1, 2, 3\}$ denote the position of $c_j$ in which this literal occurs. Add the pairs $(p^i_j, c^s_j)$ and $(u^i_j, z^s_j)$ (1 \leq k \neq s \leq 3), (p^i_j, z^s_j), (q_j, c^s_j)$ and $(u^i_j, w_j)$ to $M$.

As $M$ is a complete matching in $I$, clearly no woman in $Y \cup Z$ can be involved in a blocking pair of $M$ in $I$. Nor can a man in $P \cup U$ (since he can only potentially prefer a
woman in $Z$), nor a man in $Q$ (since he can only potentially prefer a woman in $C$, who ranks him last), nor a woman in $W$ (since she can only potentially prefer a man in $U$, who ranks him last). Now suppose that $(x_{4i+r}, c(x_{4i+r}))$ blocks $M$, where $0 \leq i \leq n-1$ and $0 \leq r \leq 3$. Let $c_j^i = c(x_{4i+r})$, where $1 \leq j \leq m$ and $1 \leq s \leq 3$. Then $(q_j, c_j^i) \in M$. If $r \in \{0, 1\}$ then $(x_{4i+r}, y_{4i+r+1}) \in M$, so that $v_i$ is false under $f$. But literal $v_i$ occurs in $c_j$, a contradiction, since literal $v_i$ was supposed to be true under $f$ by construction of $M$. Hence $r \in \{2, 3\}$ and $(x_{4i+r}, y_{4i+r}) \in M$, so that $v_i$ is true under $f$. But literal $\overline{v_i}$ occurs in $c_j$, a contradiction, since literal $\overline{v_i}$ was supposed to be true under $f$ by construction of $M$. Hence $M$ is stable in $I$.

Conversely suppose that $M$ is a complete stable matching in $I$. We form a truth assignment $f$ in $B$ as follows. For each $i \ (0 \leq i \leq n-1)$, if $M \cap (X_i \times Y_i) = T_i$, set $v_i$ to be true under $f$. Otherwise $M \cap (X_i \times Y_i) = F_i$, in which case we set $v_i$ to be false under $f$.

Now let $c_j$ be a clause in $C (1 \leq j \leq m)$. There exists some $s \ (1 \leq s \leq 3)$ such that $(q_j, c_j^s) \in M$. Let $x_{4i+r} = x(c_j^s)$ for some $i \ (0 \leq i \leq n-1)$ and $r \ (0 \leq r \leq 3)$. If $r \in \{0, 1\}$ then $(x_{4i+r}, y_{4i+r}) \in M$ by the stability of $M$. Thus variable $v_i$ is true under $f$, and hence clause $c_j$ is true under $f$, since literal $v_i$ occurs in this clause. If $r \in \{2, 3\}$ then $(x_{4i+r}, y_{4i+r+1}) \in M$ (where addition is taken modulo 4) by the stability of $M$. Thus variable $v_i$ is false under $f$, and hence clause $c_j$ is true under $f$, since literal $\overline{v_i}$ occurs in this clause. Hence $f$ is a satisfying truth assignment of $B$. \hfill \Box