A MAXIMUM STABLE MATCHING FOR THE ROOMMATES PROBLEM

JIMMY J. M. TAN*

Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

Abstract.

The stable roommates problem is that of matching n people into \( \frac{n}{2} \) disjoint pairs so that no two persons, who are not paired together, both prefer each other to their respective mates under the matching. Such a matching is called "a complete stable matching". It is known that a complete stable matching may not exist. Irving proposed an \( O(n^2) \) algorithm that would find one complete stable matching if there is one, or would report that none exists. Since there may not exist any complete stable matching, it is natural to consider the problem of finding a maximum stable matching, i.e., a maximum number of disjoint pairs of persons such that these pairs are stable among themselves. In this paper, we present an \( O(n^2) \) algorithm, which is a modified version of Irving's algorithm, that finds a maximum stable matching.

C.R. categories: F2.2, G2.1.

Key words: stable roommates problem, stable matching, maximum stable matching, algorithms.

1. Introduction.

The stable roommates problem was first described in the paper of Gale and Shapley [1]. The problem is defined as follows. There is a set \( S \) of \( n \) people. Each person \( i \) has a preference list consisting of a subset \( S_i \) of \( S - \{i\} \) and a rank ordering (most preferred first) of the persons in \( S_i \). For person \( i \), the set \( S_i \) has the meaning that the only persons he is willing to be matched with are those in \( S_i \). A complete matching \( M \) is a partition of the \( n \) persons into \( n/2 \) disjoint pairs of partners such that for every pair \( \{x, y\} \) in \( M \), \( x \) is on \( y \)'s list and \( y \) is on \( x \)'s list. A complete matching \( M \) is unstable if there are two persons who are not matched together in \( M \), but who each prefer the other to their respective partners in \( M \); such a pair is said to block the matching \( M \). A complete matching which is not unstable is called stable. The problem is to find a complete stable matching.

Gale and Shapley proposed this problem and gave the following example to show...
that a complete stable matching may not exist. In this example anyone paired with person 4 will cause instability.

<table>
<thead>
<tr>
<th>person</th>
<th>preference list</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 3 4</td>
</tr>
<tr>
<td>2</td>
<td>3 1 4</td>
</tr>
<tr>
<td>3</td>
<td>1 2 4</td>
</tr>
<tr>
<td>4</td>
<td>arbitrary</td>
</tr>
</tbody>
</table>

Fig. 1

Irving [4] proposed a polynomial-time \(O(n^2)\) in the worst case) algorithm that finds one complete stable matching if one exists, otherwise reports that none exists. Recently Gusfield and Irving published a book [31] in which they listed over a hundred research papers related to this problem. In our bibliography, we cite only a few of them which are directly relevant to us. Since there may not exist any complete stable matching, it is natural to consider the problem of finding a maximum number of disjoint pairs of persons such that these pairs are stable among themselves, i.e. no two persons, who are not paired together but have matched partners, both prefer each other to their partners under the matching. We call this "a maximum stable matching problem". In this paper, we propose an efficient algorithm to find one maximum stable matching. Our algorithm is a modification of that of Irving, and has the same \(O(n^2)\) time bound.

2. Definitions.

An instance of the stable roommates problem is specified by a set of preference lists, one for each person. Let \(S\) be a set of \(n\) persons, and \(T\) be the table of preference lists of these \(n\) persons. From now on, we assume that the table \(T\) is symmetric, i.e., person \(a\) is on \(b\)'s list if and only if \(b\) is on \(a\)'s. If person \(b\) is on the preference list of person \(a\), then we write \((a|b)\) to represent the entry \(b\) in \(a\)'s preference list.

Define \(\text{rank}(a, b) = m\), if person \(b\) occupies position \(m\) in \(a\)'s preference list. If \(\text{rank}(a, b) < \text{rank}(a, c)\), it means that person \(a\) prefers \(b\) to \(c\).

Let \(T\) be a table of preference lists. The previous definition of a complete matching being stable in \(T\) can be extended in a natural way to that of a matching (not necessarily complete) being stable in \(T\). A matching \(M\) is unstable in table \(T\) if there are two persons who are not paired together, but have matched partners in \(M\), and both prefer each other to their respective mates. A matching in table \(T\) is said to be stable, if it is not unstable. A maximum stable matching \(M\) in table \(T\) is defined to be a stable matching with the maximum number of pairs.

In the next section, we will present an algorithm for finding a maximum stable matching. Our algorithm successively deletes unnecessary entries from the table of preference lists, and locates a solution at the end. For convenience, the current set of
preference lists at any point in the algorithm is called a table. A stable matching \( M \) is said to be contained in a table \( T \), if every matched pair in \( M \) is in \( T \), i.e., in table \( T \), \( a \) is on \( b \)'s list, and \( b \) is on \( a \)'s list for each matched pair \{\( a, b \)\} in \( M \).

3. An algorithm for a maximum stable matching.

3.1 The first phase of the maximum stable matching algorithm.

We now describe an algorithm that finds a maximum stable matching. Our algorithm is divided into two phases. The fundamental idea used in Phase 1 of the algorithm is that of the "proposal and rejection" [1, 4], which is to be described as follows: If person \( a \) proposes to the current first person \( c \) on \( a \)'s list, then person \( c \) deletes (i.e., rejects) every entry \( (c|x) \) with rank\((c, a) < \) rank\((c, x)\) and its corresponding entry \( (x|c) \) from the table. An entry deleted in Phase 1 may be in some maximum stable matching. However, the following theorem shows that there exists at least one maximum stable matching contained in the resulting table. We say that a matching is stable in a table \( T \), if it is stable in the roommates instance defined by \( T \).

**Theorem 3.1.** Let \( T_0 \) be a table. Suppose that person \( c \) is the first person on \( a \)'s list in \( T_0 \), and \( x \) denotes the persons that are behind \( a \) on \( c \)'s list. Let \( T_1 \) be the table obtained from \( T_0 \) by deleting every entry \( (c|x) \) with rank\((c, a) < \) rank\((c, x)\) and its corresponding entry \( (x|c) \) from \( T_0 \). Then

(i) at least one of the maximum stable matchings in \( T_0 \) is also a stable matching in \( T_1 \);
(ii) conversely, any stable matching in \( T_1 \) is also stable in \( T_0 \).

**Proof.** (i) Let \( M_0 \) be a maximum stable matching in \( T_0 \). If the pair \( \{c, x\} \notin M_0 \) for every \( x \) that is behind \( a \) on \( c \)'s list, then the case is trivial. So we may assume that \( M_0 \) contains such a pair \( \{c, x\} \). Since \( c \) is the first person on \( a \)'s list, \( a \) must be an unmatched person in \( M_0 \), otherwise \( a \) and \( c \) block the matching. We then construct a new matching \( M_1 \) by deleting the pair \( \{c, x\} \) from \( M_0 \) and adding the pair \( \{a, c\} \) to it. It is easy to see that \( M_1 \) is also a maximum stable matching in \( T_0 \), and \( M_1 \) is contained in \( T_1 \). This proves (i).

(ii) Let \( M \) be a matching stable in table \( T_1 \). Suppose that \( M \) is not stable in \( T_0 \), then any instability must be caused by those "deleted" entries. It is a simple matter to check that these entries will not cause any instability, so (ii) follows.

So the first phase of our algorithm for finding a maximum stable matching is simply applying the proposal and rejection process, and deleting those entries behind \( x \) on \( y \)'s list from table \( T_0 \) when person \( x \) proposes to \( y \), as stated in theorem 3.1, until each person either

(i) holds a proposal;
or (ii) has an empty list.

The table obtained at the end of this phase is called the **Phase 1 table**.
PROPERTY 3.2. Let $T_1$ be a phase 1 table. Person $a$ is on $b$'s list in $T_1$ if and only if $b$ is on $a$'s list in $T_1$.

PROPERTY 3.3. Let $T_1$ be a Phase 1 table. Person $a$ is the first person on $b$'s list in $T_1$ if and only if $b$ is the last person on $a$'s list in $T_1$.

PROPOSITION 3.4. Let $T_1$ be the Phase 1 table. In table $T_1$, if every person has zero or one entry on his list, then the lists specify a maximum stable matching.

PROOF. By properties 3.2 and 3.3, if $b$ is the only person on $a$'s list in $T_1$, then $a$ is also the only person on $b$'s list in $T_1$. So $a$ and $b$ form a pair in table $T_1$. Let $\{\{a_i, b_i\} | i = 1 \text{ to } k\}$ be the set of all such pairs in $T_1$, and let $\{c_j | j = 1 \text{ to } m\}$ be the set of all persons whose lists are empty in $T_1$. Then it is obvious that $M = \{\{a_i, b_i\} | i = 1 \text{ to } k\}$ is a maximum stable matching for $T_1$. By inductively applying theorem 3.1, $M$ is also a maximum stable matching for the original table $T_0$. 

So, checking the Phase 1 table at the end of this phase,
(i) if every person has zero or one entry on his list, the lists specify a maximum stable matching;
(ii) if some of the lists contain more than one person, this brings us to the second phase of the algorithm.

It remains to describe how we deal with the Phase 1 table when some of the lists in it contain more than one person.

3.2 The second phase of the algorithm.

In the second phase of our algorithm, entries are removed from the table in a very special way. The fundamental idea is a rotation introduced by Irving [4]. We first need some definitions.

DEFINITION. Let $T$ be a table. A rotation $R$, denoted by $R = (a_1, a_2, \ldots, a_r) | (b_1, b_2, \ldots, b_s)$, is a cyclic sequence $a_1, a_2, \ldots, a_r$ of distinct people,
where $b_i$ is the first person on $a_i$'s list in $T$, and the second person on $a_{i-1}$'s, $i = 1$ to $r$ (subscripts mod. $r$). The rotation $R$ is said to be exposed in table $T$.

Example: In fig. 2(b), $(1, 2, 3) (5, 3, 4)$ is a rotation exposed in Phase 1 table.

**DEFINITION.** Let $R = (a_1, a_2, \ldots, a_r) (b_1, b_2, \ldots, b_r)$ be a rotation exposed in table $T$. Rotation $R$ is said to be eliminated from $T$, if the following entries are deleted from $T$:

(i) every entry $(b_i | x)$ with rank$(b_i, a_i) <$ rank$(b_i, x)$, $i = 1$ to $r$ (subscripts mod. $r$);

(ii) every entry $(x | b_i)$, where entry $(b_i | x)$ is described in case (i).

Example: In Fig. 2(b), after rotation $(1, 2, 3) (5, 3, 4)$ being eliminated from the table, the resulting table is as follows.

<table>
<thead>
<tr>
<th>person</th>
<th>preference list</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 4</td>
</tr>
<tr>
<td>2</td>
<td>4 5</td>
</tr>
<tr>
<td>3</td>
<td>5 1</td>
</tr>
<tr>
<td>4</td>
<td>1 2</td>
</tr>
<tr>
<td>5</td>
<td>2 3</td>
</tr>
<tr>
<td>6</td>
<td>empty</td>
</tr>
</tbody>
</table>

Fig. 3

**DEFINITION.** A table $T$ is said to be in the second phase, or in Phase 2, if

(i) the table has been subjected to Phase 1 reduction as described before;

(ii) the table has been subjected to zero or more rotation eliminations (called Phase 2 reduction).

It is not difficult to see that property 3.2 and property 3.3 also hold for a table in Phase 2. Furthermore Irving [4] proved the following proposition.

**PROPOSITION 3.5 [Irving]** Let $T$ be a table in Phase 2. If there is a person whose current list has more than one entry, then there is a rotation exposed in $T$.

In the second phase of our algorithm, we use rotation elimination to delete more entries from the table. We begin our discussion by considering the case when the elimination of a rotation makes some list empty, and study the structure of such a rotation. For the following theorem, it is helpful to use the example given in Fig. 3 to get some intuition.

**THEOREM 3.6** Let $T_2$ be a table in phase 2, and let $R = (a_1, a_2, \ldots, a_r) (b_1, b_2, \ldots, b_r)$ be a rotation exposed in $T_2$. Suppose that, on eliminating $R$ from $T_2$ some list becomes empty, then
(i) \( r = 2m + 1 \) for some \( m \), and \( b_i = a_{i+m} \) for all \( i \), \( 1 \leq i \leq r \) (subscripts mod. \( r \));

(ii) For each \( i \) there are only two entries in \( a_i \)'s list in \( T_2 \), namely \( b_i(= a_{i+m}) \) and \( b_{i+1}(= a_{i+m+1}) \);

(iii) the list of each \( a_i \), but no other list, becomes empty on eliminating \( R \).

**Proof.** On eliminating \( R \), the only lists to lose their first entry are those of the \( a_i \), so certainly no other list can become empty. Without loss of generality, we may assume that the list of \( a_1 \) becomes empty. Now \( b_2 \) is the second entry in \( a_1 \)'s list, and \((a_1 | b_2)\) can be deleted only if rank \((a_1, b_2) > \text{rank } (a_1, a_k)\) for some \( k \), where \( a_1 \) is second in the list of \( a_k \). So \( a_k = b_1 \), since \( b_1 \) is the only element for which this is true, and \( a_1 = b_{k+1} \). But \( a_1 \) is the last entry on the list of \( b_1 (= a_k) \), so this list has only two entries, namely \( a_k \) and \( a_1 \) (since \( b_1 \) is certainly on the list of both of these).

We have shown that the list of \( a_k \) contains only two entries, \( a_k = b_1 \) and \( a_1 = a_{k+1} \). Using a similar argument we can prove that \( a_{k-1} = b_r \), \( a_r = b_k \) and the list of \( a_{k-1} \) contains just two entries. By inductively applying the same argument we see that the list of \( a_{k-i} \) contains only two entries, \( a_{k-i} = b_{r-i+1} \) and \( a_{r-i+1} = b_{k-i+1} \) for \( i = 0, 1, 2, \ldots, r-1 \) (subscripts mod. \( r \)). Substituting \( i = k-1 \) into the first equality, and \( i = 0 \) into the second, we have \( a_1 = b_{r-(k-1)+1} \) and \( a_1 = b_{k+1} \). So \( r - (k-1) + 1 = k + 1 \) (mod. \( r \)), and hence \( r = 2k - 1 \) is the only possibility. Let \( m = k-1 \), then \( r = 2m + 1 \). Rewriting the equality \( a_{k-i} = b_{r-i+1} \) (replacing \( i \) by \( 1-j \)) we have \( a_{k-1-j} = b_{r-1-j+1} \) (subscripts mod. \( r \)), and \( a_{m+j} = b_j \) for \( j = 1, 2, \ldots, r \).

Consider the example given in fig. 3, if we eliminate rotation \( R = (1, 2, 3, 4, 5) | (3, 4, 5, 1, 2) \), then some lists become empty. In this example, it is straightforward to check that \( r = 2m + 1 \), where \( r = 5 \) and \( m = 2 \), and \( b_i = a_{i+m} \) for \( i = 1 \) to 5.

In the context of the theorem above, if the elimination of rotation \( R \) makes some list empty, then the lists of every person involved in this rotation become empty after eliminating it. Notice that before eliminating \( R \), any person in \( A = \{a_1, a_2, \ldots, a_r\} \) is not on the current lists of the rest of the persons. Therefore the lists of persons in \( A \) is independent of the other portion of the current table \( T_2 \). So we know that there exists no maximum stable matching contained in the resulting table after eliminating this rotation. This is because for any stable matching contained in the resulting table, one may add at least one more pair, e.g. \( \{a_1, a_2\} \) to it to have a larger stable matching in the original table. Once we detect such a rotation, we do not eliminate it, just separate the preference lists of persons in \( A = \{a_1, a_2, \ldots, a_r\} \) from the table, and treat the rest of the table as a smaller instance table (in phase 2). For convenience, such a rotation is called an *odd rotation*.

Let \( T_0 \) be the original table of preference lists, and let \( T_2 \) be a resulting table obtained from \( T_0 \) after the Phase 1 process and after eliminating zero or more rotations \( R \), i.e., \( T_2 \) is a table in Phase 2. Let \( R \) be a rotation exposed in \( T_2 \). We now consider the case that the elimination of \( R \) makes no list empty. In the following, we have a result which reveals the relationship between such an exposed rotation and a maximum stable matching. The idea of this result is inspired by [4].
THEOREM 3.7. Let $T_2$ be a table in Phase 2, and let $R = (a_1, a_2, \ldots, a_r)\rightarrow (b_1, b_2, \ldots, b_r)$ be a rotation exposed in $T_2$. Suppose that no list, which is non-empty in $T_2$, will become empty after eliminating $R$. Let $T_3$ be the resulting table obtained from $T_2$ after eliminating rotation $R$. Then

(i) at least one of the maximum stable matchings in $T_2$ is also a stable matching in $T_3$;
(ii) conversely, any stable matching in $T_3$ is also stable in $T_2$.

PROOF. (i) We first claim that, in table $T_2$, there exists a maximum stable matching in which none of $\{a_i, b_i\}$ is a matched pair, $i = 1(1)r$.

Let us accept this claim for the time being; we will prove it later. Then the entries $(a_i | b_i)$ and $(b_i | a_i)$, $i = 1(1)r$, can be deleted from the table, and the operation of proposal and rejection can be performed again. We note that $b_{i+1}$ is now the first person on $a_i$'s list. In particular, this means that no entry $(a_i | b_{i+1})$ has been deleted. For otherwise, in table $T_2$, $a_i = b_j$ for some $1 < j < r$, and the second person on $a_i$'s list is the same as the last person on $b_j$'s; then the list of $a_i$ becomes empty after eliminating $R$, giving a contradiction. Then after a sequence of proposals and rejections, we obtain a table which is exactly table $T_3$ obtained from $T_2$ after eliminating $R$. So (i) follows by inductively applying theorem 3.1.

We now prove the claim. Let $M$ be a maximum stable matching in table $T_2$. There are two cases to be considered.

(a) $a_i$ and $b_i$ are matched partners in $M$.

(b) $a_i$ and $b_i$ are not partners in $M$ for some $i$.

In both cases, we will prove that there exists another maximum stable matching in which none of $\{a_i, b_i\}$ is a matched pair for all $i$.

(a) We know that $b_i$ is the first person on $a_i$'s list and $a_i$ is the last person on $b_i$'s list, and that $a_i$ is matched with $b_i$. Let $A = \{a_1, a_2, \ldots, a_r\}$, $B = \{b_1, b_2, \ldots, b_r\}$.

If $A \cap B \neq \emptyset$, say $a_j = b_k$, since $a_j$ is matched with the first person $b_k$ on his list and $b_k$ is matched with the last person $a_k$, it is impossible that someone is matched with the first person on his list and at the same time he is also matched with the last person on his list. So we know that $A \cap B = \emptyset$. We then construct a new matching $M'$ by deleting the pair $\{a_i, b_i\}$ from $M$ and adding the pair $\{a_i, b_{i+1}\}$ to it.

Notice that $a_i$ and $b_i$ are not partners in $M'$. We claim that $M'$ is a maximum stable matching in $T_2$. Because $|M'| = |M|$ and $M'$ is a matching, we need only prove that $M'$ is stable in $T_2$. Clearly, each member $b_i$ of $B$ obtains a better partner in $M'$, from his point of view, than the one he had in $M$. The only individuals who fare worse in $M'$ than in $M$ are the members $a_i$ of $A$, so any instability in $M'$ must involve some $a_i$. Person $a_i$ is matched with the second person $b_{i+1}$ on his list and his best choice is person $b_i$. But $b_i$ prefers his partner in $M'$ to $a_i$. Therefore $M'$ is stable in $T_2$. This handles case (a).

(b) If $\{a_i, b_i\} \notin M$, then the claim is true. So, without loss of generality, we may assume that $\{a_i, b_i\} \notin M$ for $i = 1(1)k$, but $\{a_{k+1}, b_{k+1}\} \in M$, where $1 < k \leq r$. Considering the pair $\{a_{k+1}, b_{k+1}\}$ and person $a_k$, we conclude that $a_k$ must be an unmatched person in $M$. For otherwise, say $a_k$ is matched with $x$, then $x$ cannot be $b_k$ or $b_{k+1}$, since $a_k$ and $b_{k+1}$ would block the matching.
We now construct a new matching \( M' \) by deleting the pair \( \{a_k, b_{k+1}\} \) from \( M \) and adding the pair \( \{a_k, b_k+1\} \) to it. Using a similar argument as in case (a), we can conclude that \( M' \) is also a maximum stable matching in \( T_2 \). It remains to show that \( \{a_k, b_{k+1}\} \not\in M \), for \( i = 1(1)k + 1 \), i.e., \( \{a_k, b_{k+1}\} \not\in a_i, b_i \) for any \( i = 1(1)k - 1 \). (We note, however, that this may not be true if \( R \) is an odd rotation.) After establishing this fact, we may repeat the same argument, eventually obtaining a maximum stable matching in which none of \( \{a_i, b_i\} \) is a matched pair. This takes care of case (b).

Now, suppose on the contrary that \( \{a_k, b_{k+1}\} = \{a_i, b_i\} \) for some \( i = 1(1)k - 1 \), then we must have \( a_k = b_i \) and \( b_{k+1} = a_i \). So \( (b_{k+1} | a_k) = (a_i | b_i) \), which means that \( a_k \) is the first person on \( b_{k+1} \)'s list. If we eliminate \( R \) from \( T_2 \), then \( b_{k+1} \)'s list will become empty. This is because every person after \( a_k \) on \( b_{k+1} \)'s list should be deleted, and the first person \( b_i \) on \( a_i \)'s list, i.e. entry \( (b_{k+1} | a_k) \), should also be deleted. Then there is none left on \( b_{k+1} \)'s list, giving a contradiction.

(ii) Let \( M \) be a matching which is stable in \( T_3 \). We claim that \( M \) is also stable in \( T_2 \). Suppose that this is not true. We know that \( T_3 \) is a table obtained from \( T_2 \) by eliminating rotation \( R = (a_1, a_2, \ldots, a_r) | (b_1, b_2, \ldots, b_r) \). So, any instability must be caused by those "deleted" entries, i.e. there exists a person \( b_i \), and there exists another person \( x \) who is behind \( a_i \) on \( b_i \)'s list such that \( x \) and \( b_i \) block the matching \( M \). However, \( M \) is a matching in table \( T_3 \), and if \( b_i \) has a matched partner in \( M \), he must be matched with a person who is above \( x \) in his list, and \( b_i \) certainly prefers his partner to \( x \). This is a contradiction and the claim follows.

As mentioned before, if the elimination of a rotation in \( T_2 \) makes som list empty, it indicates an odd rotation. We do not eliminate any odd rotation. Suppose that the elimination of an exposed rotation in \( T_2 \) makes no list empty. To find a maximum stable matching in \( T_2 \), we only have to find a maximum stable matching in \( T_3 \) after eliminating the rotation. The reason is that a maximum stable matching in \( T_3 \) is a maximum stable matching in \( T_2 \). Repeating the operation of rotation elimination until this operation cannot be performed, then in the final table there are a collection of disjoint odd rotations, a collection of persons whose lists have one person left, and a collection of persons whose lists are empty. We now must show how we find a maximum stable matching contained in the final table.

Recall that, throughout the algorithm, the table remains symmetric. So those persons whose lists have only one entry appear in pairs. Let \( \{a_i, b_i\} | i = 1(1)k \} \) be the set of all such pairs. As for the odd rotations, we merely need to deal with one odd rotation at a time because each one is fully independent of the remainder of the final table. Considering an odd rotation, since the number of persons involved in it is odd, it is obvious that at least one person has to be excluded from a matching.

We need one more result. This result points out that there exists a complete stable matching for the rest of the persons after excluding one person from each odd rotation. According to theorem 3.6, we know that \( b_i = a_{i+m} \) (subscript mod. \( r \)), for \( i = 1(1)r \), where \( r = 2m + 1 \). Since the permutation of \( R \) is cyclic, it makes no difference to exclude one person from the matching. Without loss of generality, assume person \( a_{2m+1} \) is excluded from this odd rotation \( R \). We have the following theorem.
THEOREM 3.8. Let \( R = (a_1, a_2, \ldots, a_r) (b_1, b_2, \ldots, b_r) \) be an odd rotation in the final table. Suppose that person \( a_{2m+1} \) is excluded from the matching, where \( r = 2m + 1 \). Then matching \( M = \{ \{a_1, a_{m+1}\}, \{a_2, a_{m+2}\}, \ldots, \{a_m, a_{2m}\} \} \) is stable.

PROOF. According to the definition of a rotation, we know that all the members of \( \{a_1, a_2, \ldots, a_r\} \) in \( R \) are distinct. Note that each member of \( \{a_1, a_2, \ldots, a_r\} \) is matched with the first person on his list in the final table, and each member of \( \{a_{m+1}, a_{m+2}, \ldots, a_{2m}\} \) is matched with the second person on his list. It is obvious that \( M = \{ \{a_1, a_{m+1}\}, \{a_2, a_{m+2}\}, \ldots, \{a_m, a_{2m}\} \} \) is a matching. By theorem 3.6, we know that \( a_{2m+1}, a_1, a_2, \ldots, a_{m-1} \) are the first persons on the lists of \( a_{m+1}, a_{m+2}, a_{m+3}, \ldots, a_{2m} \) respectively. Then it is easy to check that \( M \) is stable in the final table.

Consider the example given in Fig. 3, \( R = (1, 2, 3, 4, 5) (3, 4, 5, 1, 2) \) is an odd rotation. Excluding person 5, \( \{\{1, 3\}, \{2, 4\}\} \) is a stable matching, and we shall see soon that it is also a maximum stable matching in the original table (See Fig. 2).

Before giving the formal description of Phase 2, we give some definitions assuming that the initial table of preference lists is symmetric.

**DEFINITION.** Let \( T \) be the current table at a certain point of the algorithm. The preference list of a person is said to be inactive, if

(i) the list is involved in an odd rotation which has been discovered before;
(ii) the list has only one person;
or (iii) the list is empty.

The preference list of a person is active, if it is not inactive.

**DEFINITION.** The active part of a table \( T \) is the part of the table obtained from \( T \) by deleting all the inactive lists.

If there is someone whose preference list is still active, then his list contains more than one person. Hence by proposition 3.5, there is an exposed rotation. The Phase 2 of our algorithm for finding a maximum stable matching can be stated formally as follows.

1. While the active part of the table is not empty, find an exposed rotation.
   (i) If the exposed rotation is odd, then do not eliminate it, and declare inactive the lists of all persons involved in this rotation.
   (ii) Otherwise, eliminate this rotation. If some list has only one person left on it due to the elimination, then declare this list inactive.
2. If the active part of the table is empty, go to step 3.
3. Exclude one person from each odd rotation \( R \). Suppose that \( R = (a_1, a_2, \ldots, a_r) (b_1, b_2, \ldots, b_r) \) is an odd rotation, then \( b_i = a_{i+m} \) (subscript mod. \( r \)), for \( i = 1(1)r \), where \( r = 2m + 1 \). If we exclude person \( a_{2m+1} \) from the matching, then we obtain a stable matching: \( \{\{a_1, a_{m+1}\}, \{a_2, a_{m+2}\}, \ldots, \{a_m, a_{2m}\}\} \).

Finally, the resulting table induces one maximum stable matching.
The correctness of the algorithm follows from the earlier theorems. As for time complexity, since our algorithm is basically Irving's algorithm with some modification and extension, the time bound is exactly the same as in his algorithm, namely, $O(n^2)$. The analysis is also similar to that of [4] which will not be repeated here.

4. Conclusions

A maximum stable matching problem has been defined for the stable roommates problem. The motivation of defining this problem is that there are instances of the stable roommates problem for which no complete stable matching exists. In this paper, we presented an $O(n^2)$ algorithm for finding a maximum stable matching. At the heart of this algorithm is Irving's algorithm with some modification and extension.

There are still some problems worth studying. For example, in the last step of our algorithm, we arbitrarily delete one person from each odd rotation to obtain a complete stable matching for the rest of the persons; it is not difficult to construct an example in which some persons, other than those in an odd rotation, can also be excluded to get a maximum stable matching. We would like to find a characterisation of those persons that can be excluded from a maximum stable matching.

Acknowledgement

The author wishes to express his thanks to the referee whose valuable comments improved the presentation and simplified the proofs of this paper.

REFERENCES