Journal of Heuristics
https://doi.org/10.1007/s10732-020-09464-8

A max-conflicts based heuristic search for the stable marriage problem with ties and incomplete lists

Hoang Huu Viet[1] · Nguyen Thi Uyen[1] · SeungGwan Lee[2] · TaeChoong Chung[3] · Le Hong Trang[4,5]

Received: 17 December 2019 / Revised: 25 November 2020 / Accepted: 16 December 2020

Abstract
In this paper, we propose a heuristic search algorithm based on maximum conflicts to find a weakly stable matching of maximum size for the stable marriage problem with ties and incomplete lists. The key idea of our approach is to define a heuristic function based on the information extracted from undominated blocking pairs from the men's point of view. By choosing a man corresponding to the maximum value of the heuristic function, we aim to not only remove all the blocking pairs formed by the man but also reject as many blocking pairs as possible for an unstable matching from the women's point of view to obtain a solution of the problem as quickly as possible. Experiments show that our algorithm is efficient in terms of both execution time and solution quality for solving the problem.

✉ Le Hong Trang
lhtrang@hcmut.edu.vn

Hoang Huu Viet
viethh@vinhuni.edu.vn

Nguyen Thi Uyen
uyennt@vinhuni.edu.vn

SeungGwan Lee
leesg@khu.ac.kr

TaeChoong Chung
tcchung@khu.ac.kr

1 Faculty of Information Technology, Vinh University, 182 Leduan Str., Vinh City, Nghean, Vietnam

2 Humanitas College, Kyung Hee University, 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 446-701, South Korea

3 Computer Engineering Department, Kyung Hee University, 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 446-701, South Korea

4 Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, 268 Ly Thuong Kiet, Ho Chi Minh City, Vietnam

5 Vietnam National University - Ho Chi Minh City (VNU-HCM), Ho Chi Minh City, Vietnam

Published online: 06 January 2021

Keywords Adaptive search · Heuristic search · Local search · Max-conflicts · SMTI · Undominated blocking pairs

1 Introduction

The stable marriage (SM) problem, introduced by Gale and Shapley (1962), is a well-known two-sided matching problem. An SM instance of size n consists of n men and n women in which each person ranks all members of the opposite sex in strict order of preference. The problem aims to find a one-to-one-matching between the men and the women to meet criteria such as egalitarian and sex-equal stable matchings (Nakamura et al. 1995; Viet et al. 2016, 2020). However, requiring each member to rank all members of the opposite sex in strict order is obviously unrealistic for practical applications. Therefore, several extensions of the SM problem have been proposed (Iwama and Miyazaki 2008). The first popular one is the stable marriage problem with incomplete lists (SMI) (Irving 1994; Gent et al. 2001), in which each person may rank only some members of the opposite sex, i.e. each person's preference list may be incomplete. The second popular one is the stable marriage problem with ties (SMT) (Iwama et al. 1999; Halldórsson et al. 2003b), in which each person may rank some members of the opposite sex with indifference, i.e. each person's preference list may include ties. If these extensions are combined simultaneously, then we obtain the stable marriage problem with ties and incomplete lists (SMTI) (Iwama et al. 1999; Manlove et al. 2002).

Recently, the SMTI problem has received a great deal of attention from the research community due to its important role in a wide range of applications such as the Hospitals/Residents with Ties (HRT) problem (Irving and Manlove 2009; Munera et al. 2015a; Askalidis et al. 2013), the Student-Project Allocation (SPA) problem (Abraham et al. 2003; Diebold and Bichler 2017) or the Stable Marriage and Roommates problems (Cseha and Manlove 2016; Cseha et al. 2019). With ties given in preference lists of SMTI instances, three stability criteria of a matching are defined, including *weak stability*, *strong stability*, and *super-stability* (Iwama and Miyazaki 2008; Manlove et al. 2002). Among these definitions, weak stability has received the most attention in the literature (Adil et al. 2018; Gelain et al. 2010; Manlove et al. 2002; Munera et al. 2015b).

Irving et al. (2009) showed that a weakly stable matching of an SMTI instance is found by breaking the ties in an arbitrary way and applying the Gale-Shapley algorithm (1962). Accordingly, we may obtain weakly stable matchings of different sizes. The aim of the problem is to find a stable matching with maximum size, known as MAX- SMTI problem (Gent and Prosser 2002; Manlove et al. 2002). However, the MAX- SMTI problem is NP-hard (Iwama et al. 1999; Irving et al. 2009) and therefore, finding an efficient algorithm to solve the problem of large sizes is a challenge for researchers.

In this paper, we propose a max-conflicts based heuristic search algorithm, called MCS, to solve the MAX- SMTI problem. Our key idea is that, at each iteration, MCS eliminates some blocking pairs to improve stability of an unstable matching. MCS repeats until either a perfect matching is found or a maximum number of search steps

is reached. In the latter case, the found matching is either a maximum stable matching or an unstable matching. Experimental results show that our algorithm is efficient in terms of execution time and solution quality for the MAX- SMTI problem of large sizes.

The rest of this paper is organized as follows. Section 2 presents the definitions related to the problem, Sect. 3 describes the related work, Sect. 4 presents our MCS algorithm, Sect. 5 discusses the experimental results, and Sect. 6 concludes our work.

2 Preliminaries

This section gives formal definitions related to the SMTI problem (Gelain et al. 2010; Iwama et al. 2008). An SMTI instance of size n involves a set $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ of men and a set $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$ of women in which each person ranks some members of the opposite sex in an order of preference, i.e. the preference list of each person may include ties and be incomplete. If a man $m_i \in \mathcal{M}$ and a woman $w_j \in \mathcal{W}$ rank each other in their preference lists, then we say m_i and w_j find each other *acceptable*, or (m_i, w_j) is an *acceptable* pair. We denote $r_{m_i}(w_j)$ be the rank of w_j in m_i's preference list and $r_{w_j}(m_i)$ be the rank of m_i in w_j's preference list.

A matching, M, of an SMTI instance is a set of acceptable pairs such that each person belongs to at most one pair. If a man, $m_i \in \mathcal{M}$, and a woman, $w_j \in \mathcal{W}$, form a pair $(m_i, w_j) \in M$, then we say m_i and w_j are partners in M, denoted by $M(m_i) = w_j$ and $M(w_j) = m_i$. Otherwise, we say m_i and w_j are *singles* in M, denoted by $M(m_i) = \varnothing$ and $M(w_j) = \varnothing$, respectively, and we set $r_{m_i}(\varnothing) = r_{w_j}(\varnothing) = n + 1$.

Given a matching M, a man $m_i \in \mathcal{M}$ and a woman $w_j \in \mathcal{W}$ form a *blocking pair* (m_i, w_j) for M if *(i)* m_i and w_j find each other acceptable, *(ii)* m_i is either *single* in M or strictly prefers w_j to $M(m_i)$, and *(iii)* w_j is either *single* in M or strictly prefers m_i to $M(w_j)$.

A blocking pair (m_i, w_j) dominates a blocking pair (m_i, w_k) (resp. (m_k, w_j)) from the men's (resp. women's) point of view if m_i (resp. w_j) prefers w_j (resp. m_i) to w_k (resp. m_k). A blocking pair (m_i, w_j) is undominated if there is no other blocking pairs dominating (m_i, w_j) from the men's (resp. women's) point of view.

A matching, M, in an SMTI instance is called *weakly stable* if it admits no blocking pair, otherwise, it is called *unstable*. A weakly stable matching, M, is called *perfect* if all men and women are matched in M, otherwise, it is called *non-perfect*. The size of a non-perfect matching, M, is the number of men or women matched in M.

As we will consider only weakly stable matchings, in this paper, we will simply call a weakly stable matching a stable matching and therefore, the MAX- SMTI problem is to find a stable matching of maximum size.

We consider an SMTI instance consisting of eight men and eight women with their preference lists given in Table 1. In the preference lists, for example, we write m_3: w_4 $(w_2$ $w_5)$ meaning that man m_3 strictly prefers woman w_4 to women w_2 and w_5, which are equally preferred. Some examples of matchings are as follows:

(i) Matching $M = \{(m_1, w_1), (m_2, w_5), (m_3, \varnothing), (m_4, w_6), (m_5, w_2), (m_6, w_4), (m_7, w_3), (m_8, \varnothing), (\varnothing, w_7), (\varnothing, w_8)\}$ is unstable besause it has five blocking

Table 1 An SMTI example of size 8

Men's preference list	Women's preference list
m_1: w_1	w_1: m_1 (m_5 m_6)
m_2: w_5 (w_3 w_4 w_6) (w_7 w_8)	w_2: (m_3 m_5 m_6)
m_3: w_4 (w_2 w_5)	w_3: m_6 (m_7 m_8) m_5 m_2
m_4: (w_5 w_6) w_8 w_7	w_4: m_3 (m_2 m_6 m_7) m_5
m_5: (w_1 w_3) (w_4 w_5) w_2	w_5: (m_5 m_7 m_8) (m_3 m_4) m_2
m_6: (w_4 w_7) w_1 (w_2 w_3 w_8)	w_6: m_2 m_7 (m_4 m_8)
m_7: w_4 w_6 (w_3 w_5 w_7)	w_7: (m_2 m_6) m_7 m_4
m_8: w_5 w_6 w_3	w_8: (m_2 m_4) m_6

pairs $\{(m_3, w_4), (m_3, w_5), (m_5, w_5), (m_7, w_6), (m_8, w_8)\}$. Moreover, the blocking pair (m_3, w_4) dominates the blocking pair (m_3, w_5) from the men's point of view since m_3 prefers w_4 to w_5 and the blocking pair (m_3, w_4) is undominated since there is no other blocking pairs dominating it from the men's point of view.

(ii) Matching $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_5), (m_6, w_7), (m_7, w_3), (m_8, \varnothing), (\varnothing, w_2)\}$ is stable because it admits no blocking pair, where man m_8 and woman w_2 are singles. In other words, M is a non-perfect matching of size 7.

(iii) Matching $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_2), (m_6, w_7), (m_7, w_3), (m_8, w_5)\}$ is a perfect matching.

3 Related work

In the last few years, the MAX- SMTI problem has been attracting much attention from both the Operations Research and the Artificial Intelligence communities. Iwama et al. (1999) proved that MAX- SMTI is NP-complete, while Manlove et al. (2002) showed that MAX- SMTI is NP-complete even if ties occur only on one side of preference lists and they presented a 2–approximation algorithm for the problem. Irving and Manlove (2009) introduced a (p, q)–MAX- SMTI problem, i.e. each man's preference list is of length at most p and each woman's preference list is of length at most q. They presented a $(2, \infty)$–MAX- SMTI algorithm in $O(n^{3/2}log(n))$ time for MAX- SMTI instances of size n, where $q = \infty$ means that the women's preference lists are of unbounded length. Then, they also showed that (3, 3)–MAX- SMTI is NP-hard, even if ties are on one side of preference lists only.

There are several approximation algorithms proposed to consider lower bounds for the MAX- SMTI problem. An algorithm is called r-approximation for the MAX- SMTI problem if it always finds a stable matching M with $|M| \geq |M_{opt}|/r$, where M_{opt} is a stable matching of maximum size (Király 2013). Halldórsson et al. (2003a) provided a lower bound on the approximation ratio of $\frac{21}{19}$ for MAX- SMTI by using a reduction from the minimum vertex cover problem. Iwama et al. (2004) proposed an approximation algorithm based on local search for MAX- SMTI that achieves an

approximation ratio of $(2 - c\frac{log(n)}{n})$, where c is an arbitrarily positive constant and n is the size of SMTI instances. Then, Iwama et al. (2005, 2008) improved their algorithm to achieve a $(2 - c\frac{1}{\sqrt{n}})$–approximation algorithm for MAX- SMTI, where c is a constant such that $c \leq \frac{1}{4\sqrt{6}}$. Next, the same authors improved their previous algorithms to achieve the approximation ratio of 1.875 (Iwama et al. 2007). Halldórsson et al. (2007) extended their algorithm (Halldórsson et al. 2003a) for MAX- SMTI with an approximation ratio of $13/7 (< 1.858)$ if the lengths of ties in the preference lists are limited to two. McDermid (2009) proposed a $\frac{3}{2}$–approximation algorithm that runs in $O(n^{3/2}L)$ time, where n is the sum of men and women, and L is the sum of lengths of the preference lists. While Király (2013) and Paluch (2011, 2014) modified the Gale and Shapley (1962) algorithm to achieve a $\frac{3}{2}$—approximation algorithm that runs in linear time. Unfortunately, all of the above works do not give experimental evaluations on SMTI instances.

Constraint programming approaches to solve the variants of the SM problem have also been studied by several researchers. Gent and Prosser (2002) proposed an empirical study of the MAX- SMTI problem. First, they proposed an algorithm to randomly generate SMTI instances of three parameters (n, p_1, p_2), where n is the number of men or women, p_1 is the probability of incompleteness and p_2 is the probability of ties. Then, they applied a constraint programming approach to consider the influence of parameters p_1 and p_2 to solution quality. However, their experiments ran for only SMTI instances of small size (10). Manlove and O'Malley (2005) modeled a given SMI instance in terms of a constraint satisfaction problem and applied the arc consistency propagation (Bessière and Régin 1997) to find all stable matchings.

Recently, local search approaches to deal with the MAX- SMTI problem have been applied by some researchers. Gelain et al. (2010, 2013) proposed a local search algorithm, namely LTIU, for MAX- SMTI. Starting at a randomly generated matching, LTIU tries to find a better one in the neighborhoods of the matching. If a better matching is found, the current matching is moved to the better one and LTIU repeats for the current matching. However, since the number of neighborhoods is large and the computational time to evaluate each neighborhood is $O(n^2)$, LTIU is inefficient for SMTI instances of large sizes. Munera et al. (2015b) modeled MAX- SMTI as a permutation problem and applied the adaptive search method (Codognet and Diaz 2001), called AS, to solve the problem. At each iteration, AS selects a variable with the highest error in a matching and fixes it by moving to a new matching. AS re-evaluates the new matching to check whether or not it gets stuck in a local minimum and if so, it calls a reset procedure to overcome this situation. They showed by experiments that AS is efficient in terms of execution time and is able to solve MAX- SMTI of large sizes.

4 Algorithm for SMTI

In this section, we propose a max-conflicts based heuristic search, called MCS, to solve the MAX- SMTI problem. To obtain a stable matching of maximum size, our

idea is to improve stability of an unstable matching by eliminating some blocking pairs in iterations of MCS.

4.1 Heuristic definition

Given an unstable matching M, denoted by $M = \{(\cdots), (m_i, M(m_i)), (\cdots), (M(w_j), w_j), (\cdots), (M(w_k), w_k), (\cdots)\}$, in which we assume that there exist two blocking pairs (m_i, w_j) and (m_i, w_k) for M, where (m_i, w_j) dominates (m_i, w_k) from the men's point of view. If we remove the blocking pair (m_i, w_j) for M to obtain a matching M' in which m_i is matched to w_j, both $M(m_i)$ and $M(w_j)$ become singles and the other pairs remain unchanged, i.e. $M' = \{(\cdots), (m_i, w_j), (\cdots), (M(w_j), \varnothing), (\cdots), (M(w_k), w_k), (\varnothing, M(m_i))\}$, then the blocking pair (m_i, w_k) for M' is removed. This is because if (m_i, w_k) is a blocking pair for M', then $r_{m_i}(w_k) < r_{m_i}(w_j)$ and $r_{w_k}(m_i) < r_{w_k}(M'(m_k))$. However, this is a contradiction since (m_i, w_j) dominates (m_i, w_k) from the men's point of view, i.e. $r_{m_i}(w_j) < r_{m_i}(w_k)$. As a result, when (m_i, w_j) is an undominated blocking pair (UBP) from the men's point of view for M and if we remove (m_i, w_j), then all the blocking pairs formed by man m_i are removed for M'. Likewise, if we remove an UBP (m_i, w_j) from the women's point of view, then all the blocking pairs formed by woman w_j are removed for M'.

Let $X = \{(m_i, w_j) \mid m_i \in \mathcal{M}, w_j \in \mathcal{W}\}$ denote a set of UBPs from the men's point of view for an unstable matching M. For each $(m_i, w_j) \in X$, there exist no blocking pairs dominating (m_i, w_j) from the men's point of view, meaning that m_i appears once, while w_j may appear many times in X. Let $ubp(w_j)$ be the number of UBPs formed by woman $w_j \in X$, we define a heuristic function as follows:

$$h(m_i) = n \times ubp(w_j) - r_{w_j}(m_i), \text{ for all } (m_i, w_j) \in X.$$

At each search step, we choose a pair $(m_i, w_j) \in X$ such that $h(m_i)$ has the maximum value. Since (m_i, w_j) is a UBP, (m_i, w_j) is an acceptable pair or $0 < r_{w_j}(m_i) \leq n$. This means that $h(m_i) \geq 0$ for all $(m_i, w_j) \in X$. If a pair (m_i, w_j) is chosen such that $h(m_i)$ is maximum, meaning that $n \times ubp(w_j)$ is maximum, while $r_{w_j}(m_i)$ is minimum. Moreover, since $ubp(w_j)$ is weighted with n, meaning that woman w_j making the maximum number of UBPs is selected first. Then, among pairs $(m_i, w_j) \in X$ formed by w_j, a man m_i is selected such that $r_{w_j}(m_i)$ is minimum. By removing such a selected pair (m_i, w_j) at each search step, all the blocking pairs formed by man m_i are removed. Moreover, since woman w_j most prefers m_i to the others in UBPs, if we remove (m_i, w_j), all the blocking pairs formed by w_j and men, m_k, in which w_j prefers m_i to m_k will be removed, meaning that we remove as many blocking pairs as possible from the women's point of view in the current matching.

We will show that our heuristic will return a stable matching after a finite number of steps of removing UBPs. Indeed, when we remove an UBP $(m_i, w_j) \in X$ for M to obtain a matching M', then $M(w_j)$ and $M(m_i)$ become singles in M'.

i. If w_j is single in M, i.e. $M(w_j) = \varnothing$, then m_i is matched to w_j and m_i does not form any blocking pair for M'.

ii. If w_j is not single in M, then $M(w_j)$ is single in M' and keeps single until a some matching M'' is reached, i.e. $r_{M(w_j)}(\varnothing) = n + 1$, in which:
   - If there exists no woman w_z in M'' such that $r_{w_z}(M(w_j)) < r_{w_z}(M''(w_z))$, then $M(w_j)$ keeps single and does not form any blocking pair for M''.
   - If there exists a woman w_z in M'' such that $r_{w_z}(M(w_j)) < r_{w_z}(M''(w_z))$, then $M(w_j)$ and w_z form a blocking pair for M''. After removing the blocking pair $(M(w_j), w_z)$ for M'', $M(w_j)$ is matched to w_z and all the blocking pair formed by $M(w_j)$ is removed for M''.

Consequently, if there exist blocking pairs formed by a man m_i for a matching M, meaning that there exists an UBP $(m_i, w_j) \in X$ for M. After a finite number of steps of removing UBPs, $M(w_j)$ will become single or be matched to a some woman, and both m_i and $M(w_j)$ do not form any blocking pair. This means our heuristic will return a stable matching after a finite number of steps.

We consider an SMTI example in which the preference lists given in Table 1. We assume that given a matching $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing), (m_6, w_2), (m_7, w_7), (m_8, \varnothing), (\varnothing, w_3), (\varnothing, w_5)\}$, then the set of UBPs from the men's point of view for M is $X = \{(m_2, w_5), (m_4, w_5), (m_5, w_3), (m_6, w_7), (m_8, w_5)\}$. Therefore, we have $ubp(w_3) = 1$, $ubp(w_5) = 3$, $ubp(w_7) = 1$ and $h(m_2) = 21$, $h(m_4) = 22$, $h(m_5) = 5$, $h(m_6) = 7$, $h(m_8) = 23$. Table 2 shows five cases for removing $(m_i, w_j) \in X$ to obtain a matching M' and a set X' of UBPs for M'. Obviously, if (m_8, w_5) is chosen to remove, i.e. $h(m_8)$ is maximum, then M results in a matching M' which has the smallest number of UBPs for M'.

4.2 MCS algorithm

The aim of the MAX- SMTI problem is to find a stable matching of maximum size, meaning that we have to define a function to evaluate the quality of matchings. Given a matching M, we define an evaluation function, $f(M)$, for M is as follows. If M is a stable matching, then $f(M)$ is the number of singles in M, otherwise, $f(M) = n$. As such, a stable matching M of maximum size has the smallest value of $f(M)$ and if M is perfect then $f(M) = 0$. Our MCS algorithm is shown in Algorithm 1. MCS starts to find a maximum stable matching, M_{best}, from a randomly generated matching, M (lines 1–2). At each iteration, MCS runs as follows. First, it finds a set X of UBPs for M by Algorithm 2 (line 6). Second, it checks if X is empty, i.e. M is a stable matching, then if M_{best} is worse than M in terms of the number of singles, it assigns M to M_{best} (lines 8–10). If it reaches a local minimum, it overcomes this situation by Algorithm 3 and continues the next iteration (lines 11–13), otherwise, it returns a perfect matching, M_{best}. Third, it counts the number of UBPs, $ubp(w_j)$, formed by each woman $w_j \in X$ (lines 18–20) and determines heuristic values, $h(m_i)$, for every man $m_i \in X$ (lines 21–23). Fourth, it takes a random man $m_j \in X$ with a small probability of p, or takes a man $m_j \in X$ corresponding to the maximum value of $h(m_j)$ (lines 24–28). However, if several men have the same maximum value of $h(m_j)$, it randomly picks one. Finally, it removes the UBP $(m_j, M(m_j))$ for M to obtain a new matching (line 29). MCS performs iteratively until either a perfect matching is found (line 15) or a maximum

Algorithm 1: Max-Conflicts based Heuristic Search Algorithm

Input: - An SMTI instance I of size n.
- A small probability p.
- The maximum number of iterations max_iters.

Output: A matching M.

```
1.  function Main(I)
2.      M := a randomly generated matching;
3.      M_best := M;
4.      iter := 0;
5.      while (iter ≤ max_iters) do
6.          X := Find_UBPs(M);
7.          if (X = ∅) then
8.              if (f(M_best) > f(M)) then
9.                  M_best := M;
10.             end
11.             if (f(M_best) > 0) then
12.                 M := Escape_Local_Minima(M);
13.                 continue;
14.             else
15.                 break;
16.             end
17.         end
18.         for (each m_i ∈ X) do
19.             ubp(w_j) := ubp(w_j) + 1, where (m_i, w_j) ∈ X;
20.         end
21.         for (each m_i ∈ X) do
22.             h(m_i) := n * ubp(w_j) − r_{w_j}(m_i), where (m_i, w_j) ∈ X;
23.         end
24.         if (a small probability of p) then
25.             m_j := a random man m_i ∈ X;
26.         else
27.             m_j := argmax(h(m_i)), ∀m_i ∈ X;
28.         end
29.         M := removing the bloking pair (m_j, M(m_j));
30.         iter := iter + 1;
31.     end
32.     return M_best;
33. end function
```

number of iterations is reached. In the latter case, MCS returns either a maximum stable matching or an unstable matching. It should be noted that there always exists a stable matching for an SMTI instance (Irving et al. 2009) and therefore, if MCS returns an unstable matching, then we have to increase the maximum number of iterations so that MCS returns either a non-perfect or perfect matching.

The function to determine a set X of UBPs for a matching is shown in Algorithm 2. For each man $m_i \in \mathcal{M}$, the function sorts the m_i's rank list in ascending order. Then, it considers each woman w_k in m_i's rank list such that man m_i most prefers w_k to his partner w_j, i.e. $r_{m_i}(w_k) \prec r_{m_i}(w_j)$. If (m_i, w_k) forms a blocking pair, meaning that (m_i, w_k) is an UBP, the function adds the pair (m_i, w_k) to X and repeats for the next man. We note that if a man, m_i, is being single in M then $w_j = \varnothing$ and $r_{m_i}(\varnothing) = n+1$.

Table 2 Removal of UBPs for matching M with SMTI given in Table 1

Remove	M'	X'
(m_2, w_5)	$(m_1, w_1), (m_2, w_5), (m_3, w_4), (m_4, w_8), (m_5, \varnothing), (m_6, w_2), (m_7, w_7), (m_8, \varnothing), (\varnothing, w_3), (\varnothing, w_6)$	$(m_4, w_5), (m_5, w_3), (m_6, w_7), (m_7, w_6), (m_8, w_5)$
(m_4, w_5)	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_5), (m_5, \varnothing), (m_6, w_2), (m_7, w_7), (m_8, \varnothing), (\varnothing, w_3), (\varnothing, w_8)$	$(m_5, w_3), (m_6, w_7), (m_8, w_5)$
(m_5, w_3)	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_3), (m_6, w_2), (m_7, w_7), (m_8, \varnothing), (\varnothing, w_5)$	$(m_2, w_5), (m_4, w_5), (m_6, w_7), (m_8, w_5)$
(m_6, w_7)	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing), (m_6, w_7), (m_7, \varnothing), (m_8, \varnothing), (\varnothing, w_3), (\varnothing, w_5), (\varnothing, w_2)$	$(m_2, w_5), (m_4, w_5), (m_5, w_3), (m_7, w_3), (m_8, w_5)$
(m_8, w_5)	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing), (m_6, w_2), (m_7, w_7), (m_8, w_5), (\varnothing, w_3)$	$(m_5, w_3), (m_6, w_7)$

Algorithm 2: Find a set of UBPs for a matching

Input: A matching M.
Output: a set X of UBPs.

```
1.  function Find_UBPs(M)
2.      X := ∅;
3.      for (each man m_i ∈ M) do
4.          w_j := M(m_i);                          ▷ w_j is the partner of m_i ∈ M
5.          sort m_i's rank list in ascending order;
6.          for (each w_k ∈ m_i's rank list | r_{m_i}(w_k) < r_{m_i}(w_j)) do
7.              if ((m_i, w_k) is a blocking pair) then
8.                  X := X ∪ (m_i, w_k);
9.                  break;
10.             end
11.         end
12.     end
13.     return X;
14. end function
```

In our approach, we consider two cases of local minima. The first one is when MCS gets stuck in an unstable matching by using the heuristic function, and it performs a random walk with a small probability of p to overcome this problem. The second one is when MCS gets stuck in a non-perfect matching with bad quality in terms of matching size, and we propose the function shown in Algorithm 3 to solve this problem. Specifically, when MCS found a non-perfect matching, it has no way of knowing whether or not it has reached a maximum stable matching. Therefore, we improve this matching to obtain a better one in terms of larger size. When a matching is non-perfect, the number of single men is equal to that of single women. Therefore, the ability to escape a local minimum of choosing random men is equal to that of choosing random women. To escape from a local minimum, the function checks if a probability

Algorithm 3: Escape from local minima

```
   Input: A matching M.
   Output: A matching M.
1. function Escape_Local_Minima(M)
2.     if (a probability p ≤ 0.5) then
3.         U := {m_i | M(m_i) = ∅};
4.         m_j := a random man m_i ∈ U;
5.         for (each w_k ∈ m_j 's pref. list) do
6.             if (w_k is not single in M) then
7.                 break pair (M(w_k), w_k) into two singles, M(w_k) and w_k;
8.             end
9.         end
10.    else
11.    end
12.    V := {w_i | M(w_i) = ∅};
13.    w_j := a random woman w_i ∈ V;
14.    for (each m_k ∈ w_j 's pref. list ) do
15.        if (m_k is not single in M) then
16.            break pair (m_k, M(m_k)) into two singles, m_k and M(m_k);
17.        end
18.    end
19.    return M;
20. end function
```

of $p \leq 0.5$, then it takes a random man, m_j, in the set of single men (lines 3–4). Otherwise, it takes a random woman, w_j, in the set of single women (lines 11–12). If a man m_j is chosen, the function selects every woman, w_k, in m_j's preference list and breaks pair $(M(w_k), w_k)$ into two singles, $M(w_k)$ and w_k (lines 5–9). Otherwise, if a woman w_j is chosen, the function selects every man, m_k, in w_j's preference list and breaks pair $(m_k, M(m_k))$ into two singles, m_k and $M(m_k)$ (lines 13–17). By doing so, our approach generates a new matching, in which the number of blocking pairs for the matching is much smaller than that for a randomly generated matching. This allows MCS to find a maximum stable matching much more quickly than if it were to restart from a matching generated randomly.

4.3 Example

We reconsider the SMTI instance consisting of eight men and eight women with their preference lists given in Table 1. We assume that the probability to choose a random UBP of MCS is $p = 0$ and MCS starts from a random matching $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing), (m_6, w_2), (m_7, w_7), (m_8, \varnothing), (\varnothing, w_3), (\varnothing, w_5)\}$, where $f(M) = 8$. MCS assigns M to M_{best} and runs iterations given in Table 3. At the first iteration, MCS finds a set X of UBPs for M, counts the number of blocking pairs formed by each woman $w_j \in X$, finds $h(m_i)$ for each man $m_i \in X$. Because $h(m_8)$ has the maximum value at 23, MCS removes the pair (m_8, w_5) in M to obtain a new matching $M = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing), (m_6, w_2), (m_7, w_7), (m_8, w_5), (\varnothing, w_3)\}$. MCS repeats until the fifth iteration, where

Table 3 An execution of MCS algorithm for SMTI in Table 1

Iter.	M	X	$ubp(w_j \in X)$	$h(m_i \in X)$	(m_i, w_j)
1	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing),$ $(m_6, w_2), (m_7, w_7), (m_8, \varnothing), (\varnothing, w_3), (\varnothing, w_5)$	$(m_2, w_5),$ $(m_4, w_5),$ $(m_5, w_3),$ $(m_6, w_7),$ (m_8, w_5)	$ubp(w_3) = 1$ $ubp(w_5) = 3$ $ubp(w_7) = 1$	$h(m_2) = 21$ $h(m_4) = 22$ $h(m_5) = 5$ $h(m_6) = 7$ $h(m_8) = 23$	(m_8, w_5)
2	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing),$ $(m_6, w_2), (m_7, w_7), (m_8, w_5), (\varnothing, w_3)$	$(m_5, w_3),$ (m_6, w_7)	$ubp(w_3) = 1$ $ubp(w_7) = 1$	$h(m_5) = 5$ $h(m_6) = 7$	(m_6, w_7)
3	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing),$ $(m_6, w_7), (m_7, \varnothing), (m_8, w_5), (\varnothing, w_2), \quad (\varnothing, w_3)$	$(m_5, w_3),$ (m_7, w_3)	$ubp(w_3) = 2$	$h(m_5) = 13$ $h(m_7) = 14$	(m_7, w_3)
4	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, \varnothing),$ $(m_6, w_7), (m_7, w_3), (m_8, w_5), (\varnothing, w_2)$	(m_5, w_2)	$ubp(w_2) = 1$	$h(m_5) = 7$	(m_5, w_2)
5	$(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_2),$ $(m_6, w_7), (m_7, w_3), \ (m_8, w_5)$	$\{\}$			

X is empty and $f(M) = 0$. Since $f(M_{best}) > f(M)$, M is assigned to M_{best}, i.e. $f(M_{best}) = 0$, and therefore, MCS returns a perfect matching $M_{best} = \{(m_1, w_1), (m_2, w_6), (m_3, w_4), (m_4, w_8), (m_5, w_2), (m_6, w_7), (m_7, w_3), (m_8, w_5)\}$.

5 Experiments

In this section, we present experimental results to evaluate the efficiency of our MCS algorithm. To do so, we compared the execution time and solution quality found by MCS with those found by LTIU (Gelain et al. 2010, 2013) and AS (Munera et al. 2015b). Both LTIU and AS are chosen to compare with MCS since they are local search algorithms that solve efficiently the MAX- SMTI problem. In addition, we performed experiments to consider the behaviour of MCS for SMTI instances of large sizes. We ran the experiments of MCS, LTIU and AS algorithms in Matlab R2017a software environment on a laptop computer with Core i7-8550U CPU 1.8 GHz and 16 GB RAM on Windows-10.[1]

For experiments, we used the random problem generator given in Gent and Prosser (2002) to generate SMTI instances with three parameters (n, p_1, p_2), where n is the size, p_1 is the probability of incompleteness and p_2 is the probability of ties. It should be noted that the problem generator creates SMTI instances in which the men's and women's preference lists of each instance have only acceptable pairs. For example, Table 1 shows an SMTI instance of parameters (8, 0.5, 0.5), where ties in preference lists are given in braces.

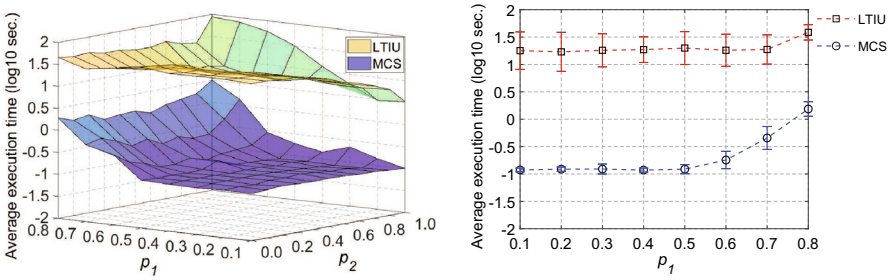
In our experiments, SMTI instances of size n are generated by letting p_1 vary in $[0.1, 0.8]$ with step 0.1 (the preference lists of SMTI instances are mostly empty for $p_1 > 0.8$), p_2 vary in $[0.0, 1.0]$ with step 0.1. Because MCS, LTIU and AS find maximum stable matchings of SMTI instances from random matchings, for each instance, we generated a random matching as an input used for both MCS, LTIU and AS algorithms. We ran MCS in which the probability to take a random UBP is $p = 0.03$.

5.1 Comparison with LTIU

We first compare the execution time and solution quality of MCS with those of LTIU. For each combination of parameters (p_1, p_2), we generated 50 SMTI instances of size 100 and averaged the results. The maximum number of iterations in MCS and LTIU is 3000.

Figure 1a shows the average execution time of MCS and LTIU for finding maximum stable matchings of the instances. The execution time of both MCS and LTIU increases significantly when p_1 increases from 0.1 to 0.8, but it increases slightly when p_2 increases from 0.0 to 1.0. Moreover, this experiment shows that MCS runs about 90 times faster than LTIU for p_1 varying from 0.1 to 0.8, as shown in Fig. 1b (thus, we use a log10 scale of execution time). These results can be explained as follows. For an SMTI instance of size n, at each iteration LTIU takes $O(n^2)$ time to determine the

[1] Source codes are available at https://github.com/vietjho/heur.



(a) MCS vs. LTIU - execution time

(b) MCS vs. LTIU for $p_2 \in [0.0, 1.0]$

Fig. 1 The average execution time of MCS and LTIU algorithms

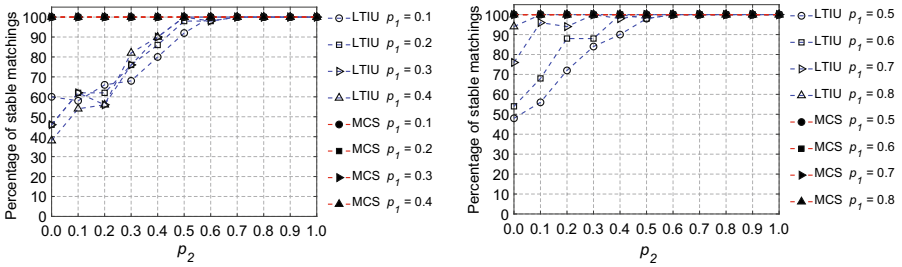
set of UBPs for a matching. By removing each blocking pair in the set of UBPs to generate a neighborhood, LTIU will achieve the set of neighborhoods of the matching. Since the cost of a matching is computed in $O(n^2)$ time, LTIU has to take $O(n^3)$ time to determine the cost of neighborhoods for finding the best matching in the set of neighborhoods of the current matching. Obviously, the larger the size of SMTI instances is, the slower LTIU runs. However, at each iteration, MCS takes only $O(n^2)$ time to determine the set of UBPs and removes only one in the set of UBPs based on the heuristic function to generate a new matching for the next iteration. This allows MCS to run much faster than LTIU.

Figure 2 shows the percentage of stable matchings found by MCS and LTIU algorithms. MCS finds 100% of stable matchings, while LTIU does not always find such matchings, especially for p_2 varying from 0.1 to 0.6. Figure 3 shows the percentage of perfect matchings found by MCS and LTIU. For p_1 varying from 0.1 to 0.5, MCS always finds 100% of perfect matchings. For p_1 varying from 0.6 to 0.8, MCS finds the percentage of perfect matchings that are higher than those found by LTIU. This can be explained as follows. When a found matching is non-perfect, LTIU applies a restart strategy to generate a new random matching for the next iteration. This means if the maximum number of iterations is very large, LTIU can find perfect matchings with a higher percentage, but it also takes much more runtime for this task. However, in these experiments, the maximum number of iterations is 3000, therefore LTIU cannot obtain a higher percentage of perfect matchings, but by such maximum number of iterations, MCS outperforms LTIU in terms of execution time and solution quality.

5.2 Comparison with AS

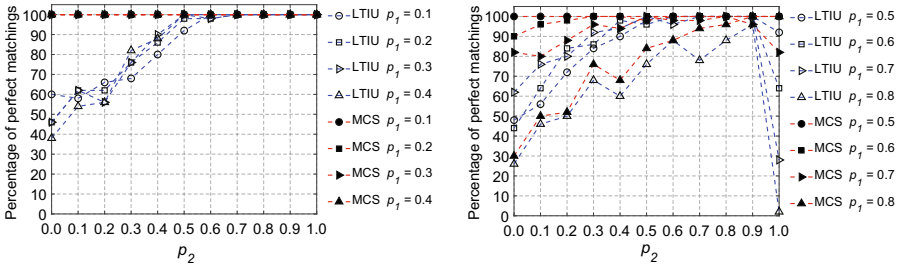
As shown in Munera et al. (2015b), SMTI instances of size 100 are easy to solve with the AS algorithm. Therefore, for each combination of parameter values (p_1, p_2), we generated 50 SMTI instances of size 500 and averaged the results to compare MCS with AS. The maximum number of iterations in MCS and AS is 3000.

Figure 4 shows the average execution time of both MCS and AS for finding maximum stable matchings of the instances. For p_2 varying from 0.0 to 0.9, MCS runs about 2 times faster than AS. When $p_2 = 1.0$, both MCS and AS run much slower



(a) MCS vs. LTIU for $p_1 \in [0.1, 0.4]$ (b) MCS vs. LTIU for $p_1 \in [0.5, 0.8]$

Fig. 2 The percentage of stable matchings found by MCS and LTIU algorithms

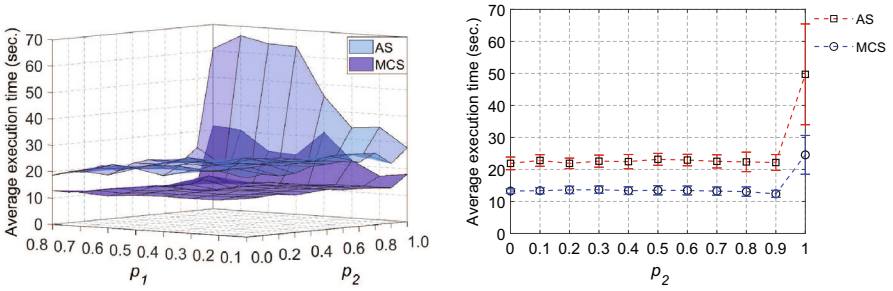


(a) MCS vs. LTIU $p_1 \in [0.1, 0.4]$ (b) MCS vs. LTIU for $p_1 \in [0.5, 0.8]$

Fig. 3 The percentage of perfect matchings found by MCS and LTIU algorithms

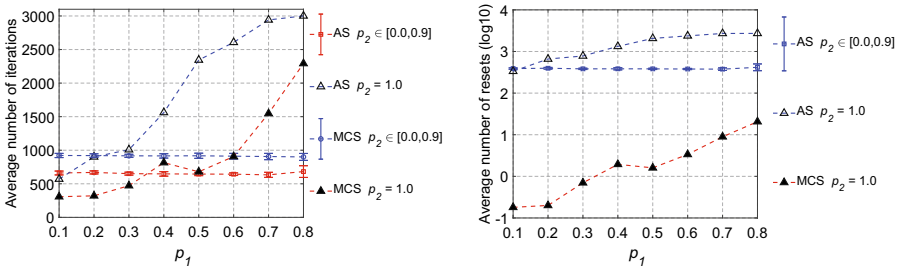
compared with p_2 varying from 0.0 to 0.9. These results can be explained as follows. At each iteration, AS takes $O(n^2)$ time to determine both a set of UBPs and the cost of a matching. After removing the worst blocking pair, i.e. a man with the highest error, in the set of UBPs to obtain a new matching, AS has to compute the cost of the new matching to compare to the previous one for checking whether or not it has reached a local minimum. This means AS takes $O(n^2)$ time to find UBPs and is added to $O(n^2)$ time to evaluate the quality of two matchings at each iteration. In contrary, as explained before, MCS takes only $O(n^2)$ time to find a set of UBPs for a matching. Therefore, MCS runs about 2 times faster than AS in general.

Figure 5a shows the average number of iterations used by MCS and AS for finding a maximum stable matching. Excepting for $p_2 = 1.0$, AS needs a fewer number of iterations (about 650 iterations) than MCS (about 850 iterations). However, the average execution time of MCS is smaller than that of AS, meaning that at each iteration AS takes much more computational time than MCS. Figure 5b shows the average number of calling the functions to escape from local minima in MCS and AS (i.e. the reset number). When p_2 varies from 0.0 to 0.9, MCS always overcomes local minima without calling the escape function. MCS only calls the escape function a few times when $p_2 = 1.0$. In contrary, AS calls the escape function about 400 times for any given p_2. This also contributes to increase the execution time of AS compared to that of MCS.



(a) MCS vs. AS - execution time (b) MCS vs. AS for $p_1 \in [0.1, 0.8]$

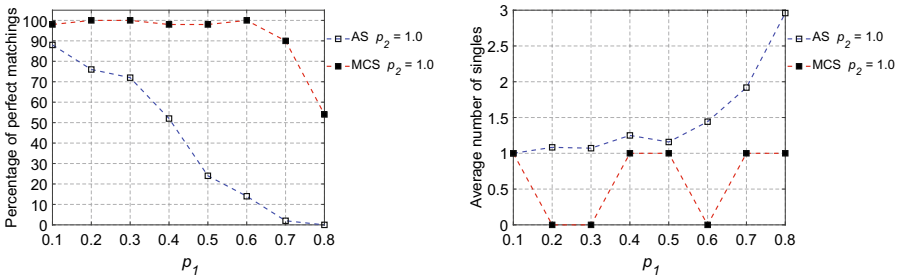
Fig. 4 The average execution time of MCS and AS algorithms



(a) MCS vs. AS - number of iterations (b) MCS vs. AS - number of resets

Fig. 5 The average number of iterations and resets used by MCS and AS algorithms

Next, we compare the quality of matchings found by MCS and AS algorithms. The experimental results show that for any given values of parameters (p_1, p_2), MCS always finds all the stable matchings, while AS still finds some unstable matchings on the SMTI instances. When p_2 varies from 0.0 to 0.9, MCS always finds all the perfect matchings, but AS still finds a few non-perfect matchings. When $p_2 = 1.0$ and p_1 varies from 0.1 to 0.7, MCS finds more than 90% of perfect matchings as shown in Fig. 6a, and the others are non-perfect matchings which have only one single as shown in Fig. 6b. In contrary, AS finds less than 90% of perfect matchings as shown in Fig. 6a, and the others are non-perfect matchings which have more than one single as shown in Fig. 6b. Especially, when $p_1 = 0.8$ and $p_2 = 1.0$, AS does not find any perfect matchings, while MCS finds about 55% of perfect matchings as shown in Fig. 6a. The experimental results demonstrate that although AS always evaluates the cost of the new matching to compare to the previous one to avoid getting stuck in a local minimum at each iteration, MCS outperforms AS in terms of finding perfect matchings. This is because, at each iteration, MCS always selects a woman making the maximum numbers of UBPs and a man in the woman's preference list so that the woman most prefers the man to the others in UBPs, then MCS removes the UBP formed by the man and woman, therefore MCS can remove as many blocking pairs as possible for a matching to achieve a new matching not only more quickly than AS but also better than that found by AS by mean of the matching cost.



(a) MCS vs. AS - % of perfect matchings

(b) MCS vs. AS - number of singles

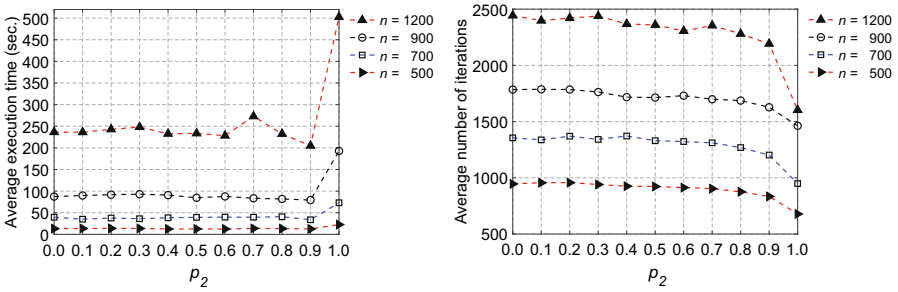
Fig. 6 The percentage of perfect matchings and the average number of singles in non-perfect matching found by MCS and AS algorithms when $p_2 = 1$

5.3 Experiments for SMTI of large sizes

In this section, we aim to evaluate the behaviour of our MCS for SMTI instances of large sizes. To do this, we randomly generated SMTI instances of sizes $n = 500, 700, 900$ and 1200 in which $p_1 = 0.5$ and p_2 varying from 0.0 to 1.0 with step 0.1. For each combination of parameters (n, p_1, p_2), we randomly generated 50 instances and averaged the results. The maximum number of iterations in MCS is 5000.

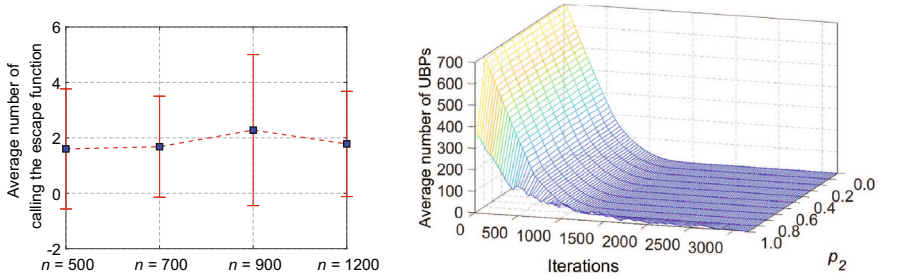
First, we consider the execution time of MCS. Figure 7a shows that, for each given value of n, the average execution time of MCS is almost the same when p_2 varies from 0.0 to 0.9. However, when $p_2 = 1.0$ the average execution time of MCS increases rapidly, especially for $n = 1200$. It should be emphasized that when the size of SMTI instances is 1200, the SMTI problem has a huge search space ($1200! = 10^{3175}$) but MCS takes only about 230 s when p_2 varies from 0.0 to 0.9 and about 500 s when $p_2 = 1.0$. Figure 7b shows the average number of iterations used by MCS. We see that the average number of iterations used by MCS decreases when p_2 increases. Moreover, MCS does not exceed 2500 iterations for $n = 1200$. It should be reminded that, as shown in Fig. 7a, when $p_2 = 1.0$, the average execution time of MCS suddenly increases, especially for $n = 1200$, while the average number of iterations decreases. This is explained as follows. In MCS, at each iteration, the time complexity to determine a set of UBPs is $O(n^2)$, while the time complexity to determine $ubp(w_j)$, as well as $h(m_i)$, is $O(n)$ only. Therefore, the time complexity of MCS depends strongly on the time complexity of determining UBPs for a matching. When $p_2 = 1.0$, i.e. ties are generated with the probability of 1.0, the rank list of all of the men consists of value 1. Therefore, the function to determine UBPs for a matching has to consider all the women, w_k, in every rank list of every man, m_i, to check whether or not man m_i prefers w_k to his partner w_j for finding a set of UBPs. This contributes significantly to the increasing in execution time when $p_2 = 1.0$.

Next, we consider local minima detected in iterations of MCS. Figure 8a shows that in a small number of iterations as shown in Fig. 7b, MCS calls a few times the escape function to overcome local minima. Figure 8b gives an example of the average number of UBPs for matchings found in iterations. The average number of UBPs does not exceed the size of SMTI instances. When p_2 varies from 0.0 to 0.9, the number of



(a) Average execution time (b) Average iteration number

Fig. 7 The average execution time and the average number of iterations used by MCS for SMTI instances of sizes 500, 700, 900 and 1200



(a) Call the escape function (b) Average number of UBPs ($n = 700, p_1 = 0.5$)

Fig. 8 The average number of calling the escape function and the average number of UBPs

UBPs decreases when the number of iterations increases, meaning that MCS does not meet any local minimum while searching for perfect matchings of SMTI instances. However, when $p_2 = 1.0$, the number of UBPs suddenly increases at a few iteration steps, meaning that MCS meets local minima and it has to call the escape function to avoid getting stuck in such situations. However, the number of UBPs for a matching generated by the escape function is much smaller than that of the matching generated randomly at the beginning of MCS and moreover, the average number to call the escape function, as shown in Fig. 8a, of MCS is just a few times. These reasons contribute significantly to accelerate the search for perfect matchings of SMTI instances.

Finally, we consider the quality of matchings found by MCS for the SMTI instances. As shown in Fig. 3, when the size of SMTI instances is 100, MCS finds 100% of perfect matchings for only p_1 varying from 0.1 to 0.5 and p_2 varying from 0.0 to 1.0. But when the size of SMTI instances is 500, MCS finds 100% of perfect matchings for p_1 varying from 0.1 to 0.8 and only p_2 varying from 0.0 to 0.9. However, the experimental results for SMTI instances of sizes 700, 900 and 1200 show that MCS always finds 100% of perfect matchings. This means that MCS finds easily perfect matchings for SMTI instances of large sizes. These results can be explained as follows. If the size of SMTI instances is small, when the values of p_1 and p_2 increase, the preference lists of men and women become sparse and contain many ties. However, if the size of SMTI instances is large, when the values of p_1 and p_2 increase, the men's and women's

preference lists contain many ties but they are not sparse as those in SMTI instances of small sizes, and therefore MCS is easily to find perfect matchings than those for SMTI instances of small sizes.

6 Conclusions

This paper proposed a max-conflicts based heuristic search algorithm, called MCS, to solve the MAX- SMTI problem. Our algorithm starts to search a solution of the problem from a random matching. At each iteration, we define a heuristic function based on undominated blocking pairs of an unstable matching from the men's point of view. Then, we remove an undominated blocking pair corresponding to the maximum value of the heuristic function to not only remove all the blocking pairs formed by the man but also reject as many blocking pairs as possible from the women's point of view. Our algorithm repeats until it finds a perfect matching or reaches a maximum number of search steps. In the latter case, the found matching is either a maximum stable matching or an unstable matching. Experiments showed that our algorithm outperforms LTIU and AS algorithms in terms of execution time and solution quality for the MAX- SMTI problem. In the future, we plan to extend the proposed approach to the Stable Matching with Uncertain Linear Preferences (Aziz et al. 2020) or Hospitals/Residents with Ties (HRT) problems (Askalidis et al. 2013).

Acknowledgements The authors are grateful to the Vietnam National Foundation for Science and Technology Development (NAFOSTED) Under Grant Number 102.01-2017.09.

References

Abraham, D.J., Irving, R.W., Manlove, D.F.: The student-project allocation problem. In: Proceedings of the 14th International Symposium, pp. 474–484. Kyoto, Japan (2003)

Adil, D., Gupta, S., Roy, S., Saurabh, S., Zehavi, M.: Parameterized algorithms for stable matching with ties and incomplete lists. Theoret. Comput. Sci. **723**(1), 1–10 (2018)

Askalidis, G., Immorlica, N., Kwanashie, A., Manlove, D.F., Pountourakis, E.: Socially stable matchings in the hospitals/residents problem. In: Proceedings of the 13th International Conference on Algorithms and Data Structures, pp. 85–96. London, ON, Canada (2013)

Aziz, H., Biró, P., Gaspers, S., de Haan, R., Mattei, N., Rastegari, B.: Stable matching with uncertain linear preferences. Algorithmica **82**(1), 1410–1433 (2020)

Bessière, C., Régin, J.C.: Arc consistency for general constraint networks: preliminary results. In: Proceedings of IJCAI '97, pp. 398–404. Morgan Kaufmann (1997)

Codognet, P., Diaz, D.: Yet another local search method for constraint solving. In: Proceedings of the International Symposium on Stochastic Algorithms, pp. 73–90. Berlin, Germany (2001)

Cseha, A., Irving, R.W., Manlove, D.F.: The stable roommates problem with short lists. Theory Comput. Syst. **63**(1), 128–149 (2019)

Cseha, A., Manlove, D.F.: Stable marriage and roommates problems with restricted edges: complexity and approximability. Discrete Optim. **20**(1), 62–89 (2016)

Diebold, F., Bichler, M.: Matching with indifferences: a comparison of algorithms in the context of course allocation. Eur. J. Oper. Res. **260**(1), 268–282 (2017)

Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Am. Math. Mon. **9**(1), 9–15 (1962)

Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Local search for stable marriage problems with ties and incomplete lists. In: Proceedings of 11th Pacific Rim International Conference on Artificial Intelligence, pp. 64–75. Daegu, Korea (2010)

Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Local search approaches in stable matching problems. Algorithms **6**(4), 591–617 (2013)

Gent, I.P., Irving, R.W., Manlove, D., Prosser, P., Smith, B.M.: A constraint programming approach to the stable marriage problem. In: Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, vol. 1, pp. 225–239. Berlin, Heidelberg (2001)

Gent, I.P., Prosser, P.: An empirical study of the stable marriage problem with ties and incomplete lists. In: Proceedings of the 15th European Conference on Artificial Intelligence, pp. 141–145. Lyon, France (2002)

Halldórsson, M.M., Iwama, K., Miyazaki, S.: Improved approximation results for the stable marriage problem. ACM Trans. Algorithms **3**(3), 1–18 (2007)

Halldórsson, M.M., Iwama, K., Miyazaki, S., Yanagisawa, H.: Improved approximation of the stable marriage problem. In: Proceedings of 11th Annual European Symposium on Algorithms, pp. 266–277. Budapest, Hungary (2003)

Halldórsson, M.M., W.Irving, R., Iwama, K., F.Manlove, D., Miyazaki, S., Morita, Y., Scott, S.: Approximability results for stable marriage problems with ties. Theor. Comput. Sci. **306**(1–3), 431–447 (2003)

Irving, R.W.: Stable marriage and indifference. Discrete Appl. Math. **48**(3), 261–272 (1994)

Irving, R.W., Manlove, D.F.: Finding large stable matchings. J. Exp. Algorithmics **14**(2), 1.2–1.2:30 (2009)

Irving, R.W., Manlove, D.F., O'Malley, G.: Stable marriage with ties and bounded length preference lists. J. Discrete Algorithms **7**(1), 213–219 (2009)

Iwama, K., Miyazaki, S.: A survey of the stable marriage problem and its variants. In: Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society, pp. 131–136. Washington, DC, USA (2008)

Iwama, K., Miyazaki, S., Morita, Y., Manlove, D.: Stable marriage with incomplete lists and ties. In: Proceedings of International Colloquium on Automata, Languages, and Programming, pp. 443–452. Prague, Czech Republic (1999)

Iwama, K., Miyazaki, S., Okamotoe, K.: A $(2 - c\frac{logN}{N})$—approximation algorithm for the stable marriage problem. In: Proceedings of the 9th Scandinavian Workshop on Algorithm Theory, pp. 349–361. Humlebek, Denmark (2004)

Iwama, K., Miyazaki, S., Yamauchi, N.: A $(2 - c\frac{1}{\sqrt{N}})$—approximation algorithm for the stable marriage problem. In: Proceedings of the 16th international conference on Algorithms and Computation, pp. 902–914. Sanya, Hainan, China (2005)

Iwama, K., Miyazaki, S., Yamauchi, N.: A 1.875: approximation algorithm for the stable marriage problem. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 288–297. New Orleans, Louisiana (2007)

Iwama, K., Miyazaki, S., Yamauchi, N.: A $(2 - c\frac{1}{\sqrt{N}})$—approximation algorithm for the stable marriage problem. Algorithmica **51**(1), 342–356 (2008)

Király, Z.: Linear time local approximation algorithm for maximum stable marriage. Algorithms **6**(1), 471–484 (2013)

Manlove, D.F., Irving, R.W., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theoret. Comput. Sci. **276**(1–2), 261–279 (2002)

Manlove, D.F., O'Malley, G.: Modelling and solving the stable marriage problem using constraint programming. In: Proceedings of the Fifth Workshop on Modelling and Solving Problems with Constraints, pp. 10–17. IJCAI'05 (2005)

McDermid, E.: A 3/2—approximation algorithm for general stable marriage. In: Proceedings of the 36th International Colloquium on Automata, Languages, and Programming, pp. 689–700. Rhodes, Greece (2009)

Munera, D., Diaz, D., Abreu, S., Rossi, F., Saraswat, V., Codognet, P.: A local search algorithm for SMTI and its extension to HRT problems. In: Proceedings of the 3rd International Workshop on Matching Under Preferences, pp. 66–77. Glasgow, UK (2015)

Munera, D., Diaz, D., Abreu, S., Rossi, F., Saraswat, V., Codognet, P.: Solving hard stable matching problems via local search and cooperative parallelization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 1212–1218. Austin, Texas (2015)

Nakamura, M., Onaga, K., Kyan, S., Silva, M.: Genetic algorithm for sex-fair stable marriage problem. In: Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on, vol. 1, pp. 509–512. Seattle, WA (1995)

Paluch, K.: Faster and simpler approximation of stable matchings. In: Proceedings of the 9th International Workshop on Approximation and Online Algorithms, pp. 176–187. Saarbrucken, Germany (2011)

Paluch, K.: Faster and simpler approximation of stable matchings. Algorithms **7**(2), 189–202 (2014)

Viet, H.H., Trang, L.H., Lee, S.G., Chung, T.C.: An empirical local search for the stable marriage problem. In: Proceedings of the 14th Pacific Rim International Conference on Artificial Intelligence - PRICAI 2016: Trends in Artificial Intelligence, pp. 556–564. Phuket, Thailand (2016)

Viet, H.H., Trang, L.H., Tuyen, L.P., Chung, T.: A shortlist-based bidirectional local search for the stable marriage problem. J. Exp. Theor. Artif. Intell. **32**(1), 147–163 (2020)