A Heuristic Repair Algorithm for the Maximum Stable Marriage Problem with Ties and Incomplete Lists

Hoang Huu Viet[1], Nguyen Thi Uyen[1], Son Thanh Cao[1], and TaeChoong Chung[2](✉)

[1] School of Engineering and Technology, Vinh University, Vinh City, Vietnam
{viethh,uyennt,sonct}@vinhuni.edu.vn

[2] Department of Computer Engineering, Kyung Hee University, Yongin, South Korea
tcchung@khu.ac.kr

Abstract. This paper proposes a heuristic repair algorithm to find a maximum weakly stable matching for the stable marriage problem with ties and incomplete lists. Our algorithm is designed including a well-known Gale-Shapley algorithm to find a stable matching for the stable marriage problem with ties and incomplete lists and a heuristic repair function to improve the found stable matching in terms of maximum size. Experimental results for large randomly generated instances of the problem showed that our algorithm is efficient in terms of both execution time and solution quality for solving the problem.

Keywords: Gale-Shapley algorithm · Heuristic repair · SMTI · Stable marriage problem

1 Introduction

The stable marriage problem with ties and incomplete lists (SMTI) [13,15] is an extension of the stable marriage (SM) problem [7]. The SMTI problem is a well-known matching problem and recently, it has been attracting much 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 [2,11,17], the Student-Project Allocation (SPA) problem [1,6] or the Stable Marriage and Roommates (SMR) problem [4,5].

An SMTI instance of size n comprises a set of men, denoted by $M = \{m_1, m_2, \cdots, m_n\}$, and a set of women, denoted by $W = \{w_1, w_2, \cdots, w_n\}$, in which each person has a preference list to rank some members of the opposite sex in an order of preference, meaning that a m_i's/w_i's preference list may include ties and be incomplete. If a man $m_i \in M$ is ranked by a woman $w_j \in W$ and vice versa, then m_i and w_j are called *acceptable* to each other, or (m_i, w_j) is an *acceptable* pair. We denote the rank of w_j in m_i's preference list by $r_{m_i}(w_j)$ and the rank of m_i in w_j's preference list by and $r_{w_j}(m_i)$. Thus, if (m_i, w_j) is an acceptable pair, then $r_{m_i}(w_j) > 0$ and $r_{w_j}(m_i) > 0$. If a man m_i strictly

G. Long et al. (Eds.): AI 2021, LNAI 13151, pp. 494–506, 2022.
https://doi.org/10.1007/978-3-030-97546-3_40

prefers a woman w_j to a woman w_k, then we denote by $r_{m_i}(w_j) < r_{m_i}(w_k)$. If a man m_i prefers a woman w_j and a woman w_k equally, then we denote by $r_{m_i}(w_j) = r_{m_i}(w_k)$. We use similar notations for the women' preference lists.

A matching Γ of an SMTI instance is a set of acceptable pairs (m_i, w_j), $(m_i, \varnothing)$, or $(\varnothing, w_j)$, meaning that each m_i or w_j belongs to at most one pair. If $(m_i, w_j) \in \Gamma$, then m_i and w_j are called partners in Γ, denoted by $\Gamma(m_i) = w_j$ and $\Gamma(w_j) = m_i$. If $\Gamma(m_i) = \varnothing$ or $\Gamma(w_j) = \varnothing$, then m_i or w_j is called *single* in Γ, respectively. A matching Γ is called *weakly stable* if it admits no *blocking pair*, where a pair (m_i, w_j) is blocking for Γ if (a) $r_{m_i}(w_j) > 0$ and $r_{w_j}(m_i) > 0$; (b) $\Gamma(m_i) = \varnothing$ or $r_{m_i}(w_j) < r_{m_i}(\Gamma(m_i))$; and (c) $\Gamma(w_j) = \varnothing$ or $r_{w_j}(m_i) < r_{w_j}(\Gamma(w_j))$. Otherwise, it is called *unstable*. The size of a weakly stable matching Γ, denoted by $|\Gamma|$, is the number of pairs $(m_i, w_j) \in \Gamma$. If $|\Gamma| = n$, then Γ is called *perfect*, otherwise, Γ is called *non-perfect*.

Irving et al. [12] showed that weakly stable matchings of an SMTI instance have different sizes. In order for every person paired, we need to find a matching that is not only weakly stable but also of maximum size. This problem is known as MAX-SMTI [10,15] and NP-hard [12,13] and therefore, finding an efficient algorithm to solve the problem of large sizes is a challenge for researchers.

In this paper, we call a weakly stable matching a stable matching. Accordingly, we propose an approximation algorithm to solve MAX-SMTI. Our idea is to apply the Gale-Shapley algorithm (GS) [7,14] for SMTI to find a stable matching. If the found matching is non-perfect, we propose a heuristic repair function to improve the matching by swapping the partners of men for single men in the matching, and then apply GS again. Our algorithm terminates when it finds a perfect matching or reaches a maximum number of iterations. Experiments show that our algorithm is efficient in terms of execution time and solution quality for solving MAX-SMTI of large sizes.

The rest of this paper is organized as follows: Sect. 2 describes the related work, Sect. 3 presents the proposed algorithm, Sect. 4 discusses the experiments, and Sect. 5 concludes our work.

2 Related Work

In the last few years, almost all algorithms proposed in the literature to solve MAX-SMTI are approximate since MAX-SMTI is NP-hard [12,13]. An algorithm is called r–approximation for MAX-SMTI if it always finds a stable matching Γ with $|\Gamma| \geq |\Gamma_{opt}|/r$, where Γ_{opt} is a stable matching of maximum size [14].

Several approximation algorithms have been extended from the GS [7] to solve MAX-SMTI. The general mechanism of these algorithms is to start from an empty matching and build a maximum stable matching through iterations. McDermid [16] proposed a 3/2–approximation algorithm that runs in $O(n^{3/2}m)$ time, where n is the sum of men and women, and m is the sum of lengths of the men's and women's preference lists. Király [14] modified GS [7] to achieve two approximation algorithms including a 3/2–approximation algorithm, namely GSA1, for SMTI where ties are allowed on one side only and a 5/3–approximation algorithm, namely GSA2, for the general case of SMTI. Paluch [19,20] gave a 3/2−approximation algorithm, namely GSM, that runs in $O(m)$ time and additionally is simpler than that of McDermid [16], where m is also the sum of the lengths of the men's and women's preference lists.

Local search has been used to solve MAX-SMTI. The general mechanism of local search-based approximation algorithms is that starting from an arbitrary matching, it improves the stability of the matching by eliminating blocking pairs through iterations until it reaches a maximum stable matching. Gelain et al. [8,9] presented a local search algorithm, namely LTIU, to deal with MAX-SMTI. Munera et al. [18] applied the adaptive search method [3], namely AS, to solve MAX-SMTI. They showed by experiments that AS outperforms LTIU in terms of execution time and solution quality. Recently, we proposed a max-conflicts-based heuristic search for MAX-SMTI [21]. Our algorithm is much more efficient than AS and LTIU in terms of execution time and solution quality for MAX-SMTI of large sizes. It should be noted that all the approaches in [8,9,18,21] used a concept of undominated blocking pair instead of blocking pair to solve MAX-SMTI. Since the computational time to determine a set of undominated blocking pairs for all men at each iteration is $O(n^2)$, these algorithms are inefficient for MAX-SMTI of large sizes.

3 Proposed Algorithm

3.1 HR Algorithm

We consider the GS given in [14] for SMTI. Given an instance I of SMTI, GS outputs a stable matching Γ_1 and we assume that Γ_1 is non-perfect, meaning that there exists some man m_i that $\Gamma_1(m_i) = \varnothing$ and m_i's preference list = $\{\}$. We consider two following cases:

Case 1: If we recover the original rank list for m_i, let m_i become active, and run GS again, then GS outputs Γ_2 which is the same as Γ_1. This is because (a) if a man $m_k \neq m_i$ and m_k was assigned to w_j in Γ_1, then m_k will keep his partner w_j in Γ_2 since there exists no man m_i such that $r_{w_j}(m_i) < r_{w_j}(m_k)$; (b) if m_i is single in Γ_1, then m_i is also single in Γ_2 since at the first run of GS, m_i was rejected by every w_j in his rank list, meaning that every w_j in m_i's rank list was assigned to some m_k or $r_{w_j}(m_k) < r_{w_j}(m_i)$ and therefore, w_j keeps her partner m_k and rejects m_i at the second run of GS.

Case 2: If we recover the original rank list for m_i, let w_j be one of the women in m_i's rank list so that either $r_{m_i}(w_j) \leq r_{m_k}(w_j)$ or $r_{w_j}(m_i) = r_{w_j}(m_k)$, where $m_k = \Gamma_1(w_j)$, then if we swap m_i for m_k in Γ_1, i.e. (a) $\Gamma_1(m_k) = \varnothing$; (b) $\Gamma_1(m_i) = w_j$; (c) delete w_j from m_k's rank list; (d) let m_k be active; and run GS again with Γ_1 as an input, then GS outputs Γ_2, in which $\Gamma_2(m_i) = w_j$ and m_k may be assigned to some woman in his rank list. If so, we have $|\Gamma_2| > |\Gamma_1|$. This is our idea to improve a stable matching in terms of maximum size.

Our heuristic repair algorithm, so called HR, to solve MAX-SMTI is shown in Algorithm 1. We call a `repair`(m_i, m_k) a procedure consisting of (a) $\Gamma(m_i) := w_j$, where $w_j = \Gamma(m_k)$, i.e. (m_i, w_j) becomes a pair; (b) $\Gamma(m_k) := \varnothing$, i.e. m_k

Algorithm 1: HR Algorithm

```
1.  function Main($I$)
2.      for ($each\ m_i \in M$) do
3.          $\Gamma(m_i) := \varnothing$;
4.          $a(m_i) := 1$;                         ▷ assign $m_i$ to be active
5.          $c(m_i) := 0$;       ▷ assign a count variable of $m_i$ to zero
6.      end
7.      $iter := 1$;
8.      while $iter \leq max\_iters$ do
9.          $m_i :=$ some man is active, i.e. $a(m_i) = 1$;   ▷ take an active man $m_i$
10.         if $there\ exists\ no\ active\ man$ then
11.             if $\Gamma\ is\ perfect$ then break;
12.             $iter := iter + 1$;
13.             $\Gamma :=$ improve($\Gamma$);
14.             continue;
15.         end
16.         if $m_i$'s rank list is empty then
17.             $a(m_i) := 0$;                     ▷ assign $m_i$ to be inactive
18.             $c(m_i) := c(m_i) + 1$;     ▷ increase the count variable of $m_i$
19.             continue;
20.         end
21.         if $there\ exists\ a\ single\ woman\ w_j\ to\ whom\ m_i\ prefers\ most$ then
22.             $\Gamma(m_i) := w_j$;                  ▷ $m_i$ becomes engaged to $w_j$
23.             $a(m_i) := 0$;
24.         else
25.             $w_j :=$ a woman to whom $m_i$ prefers most;
26.             $m_k := \Gamma(w_j)$;
27.             if $there\ exists\ a\ single\ w_t\ that\ r_{m_k}(w_t) = r_{m_k}(w_j)$ then
28.                 repair($m_i, m_k$);
29.             end
30.             if $\Gamma(m_i) = \varnothing\ and\ r_{w_j}(m_i) < r_{w_j}(m_k)$ then
31.                 repair($m_i, m_k$);
32.                 $r_{m_k}(w_j) := 0$;        ▷ delete $w_j$ from $m_k$'s rank list
33.             else
34.                 $r_{m_i}(w_j) := 0$;        ▷ delete $w_j$ from $m_i$'s rank list
35.             end
36.         end
37.     end
38.     return $\Gamma$;
39. end function
```

becomes single; (c) $a(m_i) := 0$, i.e. m_i is inactive; and (d) $a(m_k) := 1$, i.e. m_k becomes active again. At the beginning, HR creates a matching Γ of single men for each $m_i \in M$, sets each m_i to be active, and assigns a count variable for each m_i to zero (lines 2–6). At each iteration, if HR does not find any active man m_i, then it improves the matching Γ to obtain a better one in terms of maximum size (lines 10–15), otherwise, it runs GS to find a stable matching for SMTI (lines 9, 21–36). In the former case, HR checks if Γ is perfect, then it returns Γ, otherwise, it improves $|\Gamma|$ by calling Algorithm 2 and starts the next iteration. In the latter case, HR checks if m_i's rank list becomes empty (i.e. $r_{m_j}(w_j) = 0, \forall w_j \in W$), then it assigns m_i to be inactive, increases the count variable $c(m_i)$ of m_i's exhaustive search, and starts the next iteration. Otherwise, m_i proposes a single woman w_j to whom he prefers most. If there exists a such w_j, then w_j is assigned to m_i. However, if there exists no such w_j, meaning that w_j has a partner m_k. Accordingly, w_j is assigned to m_i if either m_k has a single woman w_t that $r_{m_k}(w_t) = r_{m_k}(w_j)$ or w_j prefers m_i to m_k. If w_j is assigned to m_i, then m_i becomes inactive (i.e. $a(m_i) = 0$), otherwise, m_i deletes w_j from his rank list (i.e. $r_{m_i}(w_j) = 0$). If w_j rejects m_k to be assigned to m_i, then m_k becomes active and it deletes w_j from his rank list, except m_k has a single woman w_t that $r_{m_k}(w_t) = r_{m_k}(w_j)$.

The function to improve $|\Gamma|$ is shown in Algorithm 2. For each single man $m_i \in M$, since m_i is single, meaning that it is rejected by all women in his rank list or m_i's rank list becomes empty, he first recovers his original rank list. Next, m_i finds a set of women, w_j, in his rank list such that $r_{m_i}(w_j) \leq r_{m_k}(w_j)$ or $r_{w_j}(m_i) = r_{w_j}(m_k)$, where $m_k = \Gamma(w_j)$ (lines 5–10). If there exists no such w_j, the function continues for the next single man in M. Otherwise, a woman w_j corresponding to the minimum value of $h(w_j)$ is chosen to assign to m_i and m_k, the previous partner of w_j, deletes w_j from his rank list. By doing so, m_k has opportunities to be assigned to the other women in his rank list in the next iterations of HR. It should be noted that a woman w_j is chosen such that $h(w_j)$ is minimum, meaning that *(i)* m_k has the maximum number of women w_t that $r_{m_k}(w_t) = r_{m_k}(w_j)$; *(ii)* w_j ranks m_i closest to m_k; and *(iii)* $c(m_k)$ is minimum.

3.2 Example

Considering an SMTI instance consists of eight men and eight women with their preference lists given in Table 1, where ties in the men's and women's preference lists are given in brackets. HR runs as follows:

(1) HR runs the first times of GS (lines 9, 21–36) and yields a stable matching $\Gamma = \{(m_1, w_3), (m_2, \varnothing), (m_3, w_8), (m_4, w_5), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$ after 11 iterations. At the 12th iteration, since there exists no active man and $|\Gamma| = 7$, the function `improve`(Γ) is called to improve $|\Gamma|$. Specifically, since m_2 is single, it recovers its original rank list. Next, m_2 finds w_5 to be a candidate, since w_5 has a partner m_4, it rejects m_4 to assign to m_2 and m_4 deletes w_5 in his rank list. So, the function returns $\Gamma = \{(m_1, w_3), (m_2, w_5), (m_3, w_8), (m_4, \varnothing), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$.

(2) HR runs the second times of GS and results in $\Gamma = \{(m_1, w_3), (m_2, w_5), (m_3, w_8), (m_4, \varnothing), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$ at the 14th iteration. At the 15th iteration, since there exists no active man and $|\Gamma| = 7$, the function `improve`(Γ) is called to improve $|\Gamma|$ again. Specifically, since m_4 is single, it recovers its original rank list. Next, m_4 finds w_8 to be a candidate, since w_8 has

Algorithm 2: Improve a stable matching Γ

```
1.  function Improve($\Gamma$)
2.      for each single man $m_i \in M$ do
3.          recover $m_i$'s original rank list;
4.          $X := \{\}$;
5.          for each $w_j \in m_i's$ rank list do
6.              $m_k := \Gamma(w_j)$;
7.              if $r_{m_i}(w_j) \leq r_{m_k}(w_j)$ or $r_{w_j}(m_i) = r_{w_j}(m_k)$ then
8.                  $X := X \cup \{w_j\}$;                ▷ $w_j$ is a candidate for $m_i$
9.              end
10.         end
11.         if $X$ is empty then continue;
12.         for each $w_j \in X$ do
13.             $m_k := \Gamma(w_j)$;
14.             $k :=$ number of $w_t$ in $m_k$'s rank list, where $r_{m_k}(w_t) = r_{m_k}(w_j)$;
15.             $h(w_j) := 1/k + (r_{w_j}(m_i) - r_{w_j}(m_k)) \times (1 - c(m_k))$;
16.         end
17.         $w_j := argmin(h(w_j)), \forall w_j \in X$;
18.         repair($m_i, m_k$), where $m_k := \Gamma(w_j)$;
19.         $r_{m_k}(w_j) := 0$;                ▷ delete $w_j$ from $m_k$'s rank list
20.     end
21.     return $\Gamma$;
22. end function
```

Table 1. An SMTI instance of size 8

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

a partner m_3, it rejects m_3 to assign to m_4 and m_3 deletes w_8 in his rank list. So, the function yields $\Gamma = \{(m_1, w_3), (m_2, w_5), (m_3, \varnothing), (m_4, w_8), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$.

(*3*) HR runs the third times of GS and results in a perfect matching $\Gamma = \{(m_1, w_3), (m_2, w_5), (m_3, w_7), (m_4, w_8), (m_5, w_2), (m_6, w_6), (m_7, w_1), (m_8, w_4)\}$ of size 8 at the 17th iteration.

It should be noted that in this example, GS finds a stable matching $\Gamma = \{(m_1, w_3), (m_2, \varnothing), (m_3, w_7), (m_4, w_5), (m_5, w_2), (m_6, w_6), (m_7, w_8), (m_8, w_1)\}$ of size 7 and GSA2 finds a stable matching $\Gamma = \{(m_1, w_3), (m_2, \varnothing), (m_3, w_8),$

(m_4, w_5), (m_5, w_2), (m_6, w_6), (m_7, w_1), $(m_8, w_4)\}$ of size 7. Although GSA2 improves GS but it gets stuck at the 7th iteration, where m_2 becomes inactive forever, and so m_2 is a single man.

4 Experiments

In this section, we present experiments to evaluate the performance of our HR algorithm. To do so, we chose GSA2 [14] to compare its performance with that of HR since both GSA2 and HR are improved based on GS [14]. We implemented HR and GSA2 by Matlab R2017b software on a laptop computer with Core i7-8550U CPU 1.8 GHz and 16 GB RAM, running on Windows 10. The maximum number of iterations used in HR is 50.

Datasets. We used the random problem generator given in [10] 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. Since stable matchings of SMTI instances include acceptable pairs and singles, we generated SMTI instances that the men's and women's preference lists of each instance have only acceptable pairs.

4.1 Comparison of Solution Quality

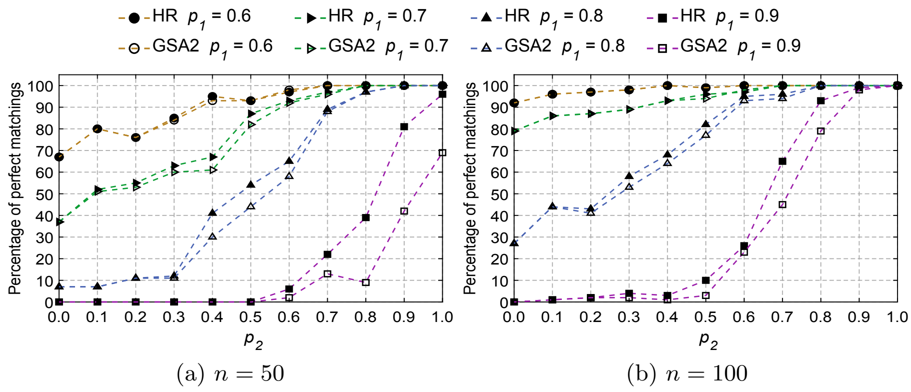
This section presents our experimental results in comparing the percentage of perfect matchings found by HR with that found by GSA2.

Experiment 1. In this experiment, we chose $n \in \{50, 100, 150, 200\}$, let $p_1 \in \{0.1, 0.2, \cdots, 0.9\}$ and $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$. For each combination of parameters (n, p_1, p_2), we generated 100 SMTI instances, ran HR and GSA2 on the generated instances. Our experimental results show that when $p_1 \in \{0.1, 0.2, \cdots, 0.5\}$ and $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$, both HR and GSA2 find 100% of perfect matchings, so we do not show the experiment results here. Figure 1 shows the percentage of perfect matchings found by HR and GSA2. From the experimental results, we can give some remarks as follows:

(*1*) The percentage of perfect matchings found by HR is higher than that found by GSA2 for cases of (*i*) $n = 50$ and $p_1 \in \{0.7, 0.8, 0.9\}$; (*ii*) $n = 100$ and $p_1 \in \{0.8, 0.9\}$; and (*iii*) $n \in \{150, 200\}$ and $p_1 = 0.9$. This means when each person ranks fewer members of the opposite sex, HR is more efficient than GSA2 in finding perfect matchings for SMTI, especially for $p_1 \in \{0.8, 0.9\}$. When n increases, meaning that each person ranks many members of the opposite sex, the percentage of perfect matchings found by HR and GSA2 increases, i.e. both HR and GSA2 find easier perfect matchings.

(*2*) When p_1 increases, the percentage of perfect matchings found by HR and GSA2 decreases since the number of acceptable pairs in the men's and women's preference lists decreases, making more difficult for finding perfect matchings.

(*3*) When p_2 increases, the percentage of perfect matchings found by HR and GSA2 increases since at the same p_1 value, the number of ties in the men's and women's preference lists increases, making easier for finding perfect matchings.



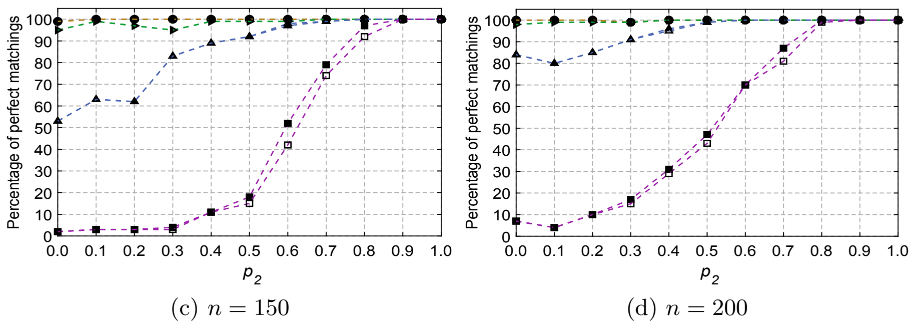


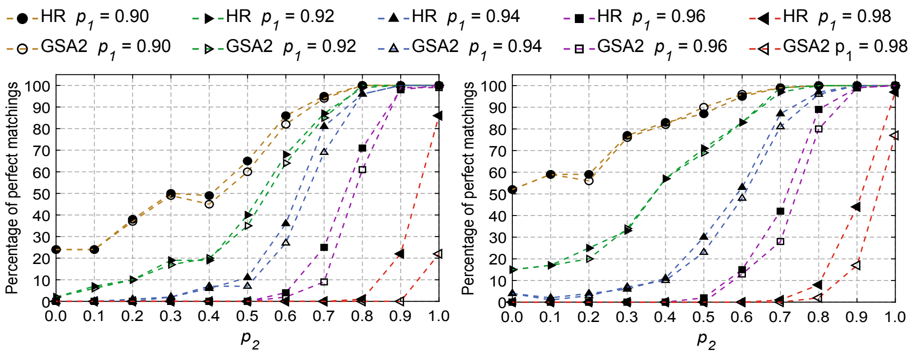
Fig. 1. Percentage of perfect matchings found for $n \in \{50, 100, 150, 200\}$

Experiment 2. In the above experiment, for $p_1 \in \{0.1, 0.2, \cdots, 0.8\}$ and when n increases, both HR and GSA2 find easy perfect matchings since the number of acceptable pairs in men's and women's rank list increases. Therefore, for example $n = 200$, the comparison of the percentage of perfect matchings found by HR and GSA2 is useless. In this experiment, we chose $n \in \{300, 400\}$, let $p_1 \in \{0.90, 0.92, \cdots, 0.98\}$ and $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$. Figure 2 shows the results of this experiment. Again, we see that when p_1 increases, the percentage of perfect matchings found by HR and GSA2 decreases and when p_2 increases, the percentage of perfect matchings found by HR and GSA2 increases. However, HR outperforms GSA2 in terms of finding perfect matchings for SMTI.

4.2 Comparison of Execution Time

In the above experiments where n is small, the average execution time of HR and GSA2 is very small and almost the same and therefore, the comparison of the execution time of these algorithms is meaningless.

Experiment 3. In this experiment, to compare the execution time of HR and GSA2 more precisely, we chose $n \in \{1000, 2000\}$, let $p_1 \in \{0.1, 0.2, \cdots, 0.9\}$



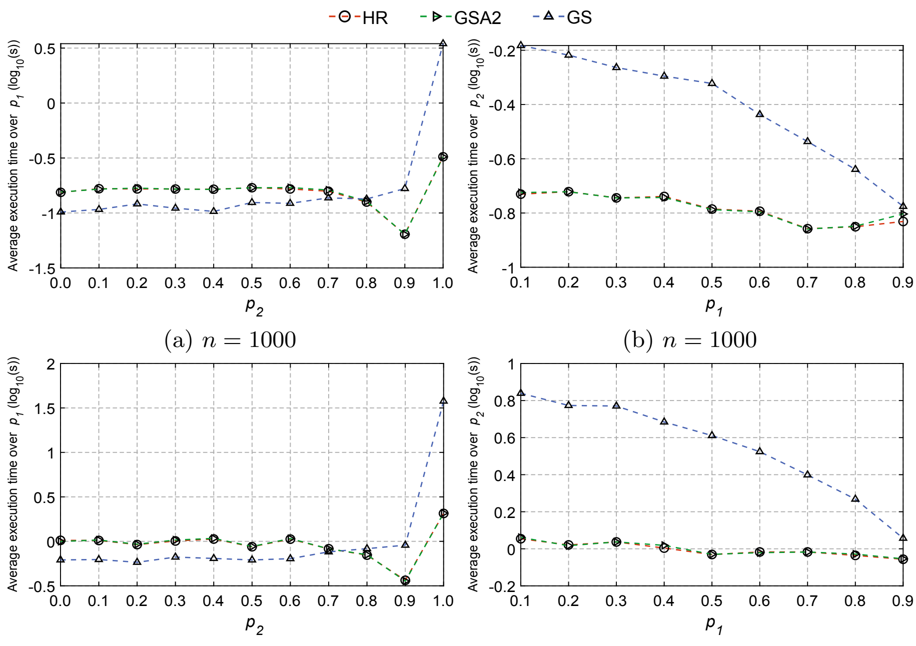
(a) $n = 300$ (b) $n = 400$

Fig. 2. Percentage of perfect matchings found for $n = \{300, 400\}$

and $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$. Since both HR and GSA2 are based on GS [9,14], we implemented GS to compare the execution time of GS with that of HR and GSA2. For each combination of parameters (n, p_1, p_2), we generated one SMTI instance, ran HR, GSA2, and GS on the generated instances. Figure 3 shows the average execution time of HR, GSA2, and GS for finding perfect matchings. We see that the execution time of HR is approximately equal to that of GSA2. When p_2 increases from 0.0 to 0.8, the execution time of both HR and GSA2 is almost unchanged, but larger than that of GS. When $p_2 = 0.9$, the execution time of both HR and GSA2 significantly decreases, but that of GS slightly increases. When $p_2 = 1.0$, the execution time of HR, GSA2 and GS increases. When p_1 increases from 0.1 to 0.9, the execution time of both HR and GSA2 is almost unchanged, while that of GS significantly decreases. It should be emphasized that when $n = 2000$, SMTI has a huge search space ($2000! \simeq 10^{5735}$ matchings), but HR runs about $10^0 = 1.0$ seconds for $p_2 \leq 0.9$ and about $10^{0.3} = 1.99$ seconds for $p_2 = 1.0$.

Experiment 4. In Experiment 3, when $p_1 \in \{0.1, 0.2, \cdots, 0.9\}$, both HR and GSA2 find 100% of perfect matchings. This may result in the execution time of HR approximately equal to that of GSA2. In this experiment, we chose n and p_2 as in Experiment 3, but let $p_1 \in \{0.91, 0.92, \cdots, 0.99\}$. Figure 4 shows the average execution time of HR, GSA2, and GS for finding perfect matchings. Again, we see that the execution time of HR is approximately equal to that of GSA2. When p_2 increases from 0.0 to 1.0, the execution time of both HR and GSA2 is decreases, while that of GS increases. When p_1 increases from 0.91 to 0.99, the execution time of HR is almost unchanged, while that of GSA2 and GS decreases. It should be noted that (i) when $n = 1000$, HR and GSA2 find 72% and 67% of perfect matchings, respectively; and (ii) when $n = 2000$, HR and GSA2 find 90% and 87% of perfect matchings, respectively.

As we mentioned above, HR consists of GS to find a stable matching and a heuristic function to maximize the matching found by GS, however, when

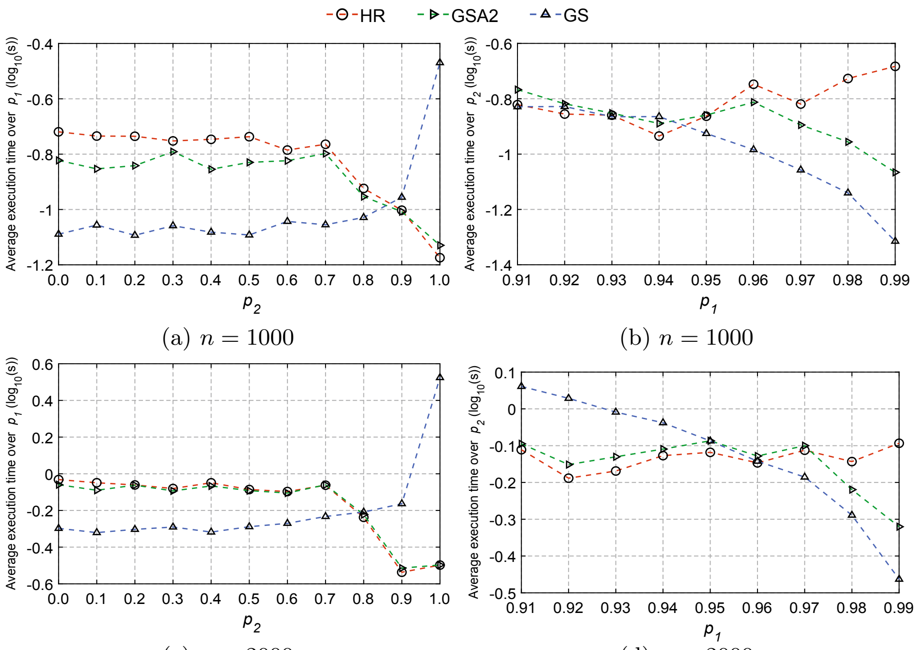


(a) $n = 1000$ (b) $n = 1000$

(c) $n = 2000$ (d) $n = 2000$

Fig. 3. Execution time for finding perfect matchings, where $p_1 \in [0.1, 0.2, \cdots, 0.9]$

$p_2 \in \{0.9, 1.0\}$ or p_1 is small, the execution time of HR is smaller than that of GS. This is because at each iteration of GS, each single man m_i proposes a woman w_j to whom he prefers most. If w_j has a partner m_k and $r_{w_j}(m_k) < r_{w_j}(m_i)$, then m_i is rejected by w_j. When $p_2 = 0.9$, each man ranks women almost equally (i.e. rarely $r_{w_j}(m_i) < r_{w_j}(m_k)$), and when $p_2 = 1.0$, each man ranks women equally (i.e. $r_{w_j}(m_i) = r_{w_j}(m_k)$), and vice versa. This means that m_i has to propose the next woman to whom he prefers most at the next iterations. If every woman w_j in m_i's preference list has a partner, then m_i has to propose every w_j and he is rejected by w_j, i.e. m_i becomes a single. In contrary, at each iteration of HR, each single man m_i proposes a woman w_j to whom he prefers most. Then, there are two cases: (i) if there exists a single woman w_j in the set of the women to whom m_i prefers equally, then w_j is assigned to m_i (lines 21–23 in HR); (ii) If w_j has a partner m_k, and if there exists a single woman w_t that $r_{m_k}(w_t) = r_{m_k}(w_j)$, then w_j is assigned to m_i (lines 27–29 in HR) and w_t has a chance to assign to m_k when m_k proposes w_t at the next some iteration. By doing so, m_i do not find the next woman to whom he prefers most at the next iterations. Obviously, when p_1 increases, each man ranks fewer women in his preference list and therefore, HR runs much faster than GS when $p_2 \in \{0.9, 1.0\}$ or p_1 is small.



(a) $n = 1000$ (b) $n = 1000$

(c) $n = 2000$ (d) $n = 2000$

Fig. 4. Execution time for finding perfect matchings, where $p_1 \in [0.91, 0.92, \cdots, 0.99]$

5 Conclusions

This paper proposed a heuristic repair algorithm, namely HR, to solve the MAX-SMTI problem. HR is designed including a well-known GS algorithm [9,14] to find a stable matching for the SMTI problem and a heuristic repair function to improve the quality of the found stable matching in terms of maximum size. The experimental results for large randomly generated instances of SMTI showed that HR outperforms GSA2 [14] in terms of solution quality for finding perfect matchings of SMTI problem. In the future, we plan to extend the proposed approach to the Hospitals/Residents with Ties problem [11,17] and the Student-Project Allocation problem [1,6].

Acknowledgment. This research is funded by the Basic Science Research Program through the National Research Foundation of Korea under grant number NRF-2020R1F1A1050014.

References

1. Abraham, D.J., Irving, R.W., Manlove, D.F.: The student-project allocation problem. In: Ibaraki, T., Katoh, N., Ono, H. (eds.) ISAAC 2003. LNCS, vol. 2906, pp. 474–484. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-24587-2_49

2. Askalidis, G., Immorlica, N., Kwanashie, A., Manlove, D.F., Pountourakis, E.: Socially stable matchings in the hospitals/residents problem. In: Dehne, F., Solis-Oba, R., Sack, J.-R. (eds.) WADS 2013. LNCS, vol. 8037, pp. 85–96. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40104-6_8
3. Codognet, P., Diaz, D.: Yet another local search method for constraint solving. In: Steinhöfel, K. (ed.) SAGA 2001. LNCS, vol. 2264, pp. 73–90. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45322-9_5
4. Cseh, Á., Irving, R.W., Manlove, D.F.: The stable roommates problem with short lists. Theory Comput. Syst. **63**(1), 128–149 (2017). https://doi.org/10.1007/s00224-017-9810-9
5. Cseha, A., Manlove, D.F.: Stable marriage and roommates problems with restricted edges: complexity and approximability. Discrete Optimizat. **20**(1), 62–89 (2016)
6. 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)
7. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Am. Math. Mon. **9**(1), 9–15 (1962)
8. Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Local search for stable marriage problems with ties and incomplete lists. In: Zhang, B.-T., Orgun, M.A. (eds.) PRICAI 2010. LNCS (LNAI), vol. 6230, pp. 64–75. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15246-7_9
9. 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)
10. 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, July 2002
11. Irving, R.W., Manlove, D.F.: Finding large stable matchings. J. Experiment. Algorithmics **14**(2), 1.2–1.2:30 (2009)
12. Irving, R.W., Manlove, D.F., O'Malley, G.: Stable marriage with ties and bounded length preference lists. J. Discret. Algorithms **7**(1), 213–219 (2009)
13. Iwama, K., Miyazaki, S., Morita, Y., Manlove, D.: Stable marriage with incomplete lists and ties. In: Wiedermann, J., van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 443–452. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48523-6_41
14. Király, Z.: Linear time local approximation algorithm for maximum stable marriage. Algorithms **6**(1), 471–484 (2013)
15. 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)
16. McDermid, E.: A 3/2-approximation algorithm for general stable marriage. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 689–700. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02927-1_57
17. 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, United Kingdom, April 2015
18. 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, January 2015

19. 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, September 2011
20. Paluch, K.: Faster and simpler approximation of stable matchings. Algorithms **7**(2), 189–202 (2014)
21. Viet, H.H., Uyen, N.T., Lee, S.G., Chung, T.C., Trang, L.H.: A max-conflicts based heuristic search for the stable marriage problem with ties and incomplete lists. J. Heuristics **27**(3), 439–458 (2021). https://doi.org/10.1007/s10732-020-09464-8