MINISTRY OF EDUCATION AND TRAINING

VIETNAM ACADEMY OF SCIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY SCIENCE AND TECHNOLOGY

NGUYEN THI UYEN

A RESEARCH ON VARIANTS OF THE STABLE MARRIAGE PROBLEM BASED ON THE HEURISTIC APPROACH

Major: **Computer Science**

Major code: **9 48 01 01**

SUMMARY OF COMPUTER DOCTORAL THESIS

Ha Noi - 2023

The thesis has been completed at: Graduate University of Science and Technology- Vietnam Academy of Science and Technology

Supervisor 1: Assoc.Prof. PhD. Hoang Huu Viet

Supervisor 2: Assoc.Prof. PhD. Nguyen Long Giang

Reviewer 1: Assoc.Prof. PhD. Pham Van Hai

Reviewer 2: Assoc.Prof. PhD. Bui Thu Lam

Reviewer 3: Assoc.Prof. PhD. Le Hoang Sơn

This thesis was defended in front of committee of Graduate University of Science and Technology – Vietnam Academy of Science and Techonology at 14 pm, May 29th, 2023

The thesis can be found at:

- The Library of Graduate University of Science and Technology
- Vietnam National Library

Ha Noi - 2023

LIST OF PUBLICATIONS

1. Publications used in the Thesis

[A.1]Hoang Huu Viet, **Nguyen Thi Uyen**, SeungGwan Lee,TaeChoong Chung, and Le Hong Trang, "*A Max-Conflicts based Heuristic Search for the Stable Marriage Problem with Ties and Incomplete Lists*", Journal of Heuristics (**SCIE - Q2**), vol. 27, no.3, pp. 439–458, 2021.

[A.2]Hoang Huu Viet, **Nguyen Thi Uyen**, Cao Thanh Sơn, and TaeChoong Chung: *A Heuristic Repair Algorithm for the Maximum Stable Mar- riage Problem with Ties and Incomplete Lists*, in Proceedings of the 34^{th} Australasian Joint Conference on Artificial Intelligence 2022 (AI 2022), Sydney, Australia, Feb.2-4, 2022, pp.494-506, Lecture Notes in Artificial Intelligence 13151 (**SCOPUS**), Springer, ISBN 978-3-030- 97545-6.

[A.3] **Nguyen Thi Uyen**, Nguyen Long Giang, Nguyen Truong Thang, and Hoang Huu Viet: *A min-conflicts algorithm for maximum stable match- ings of the hospitals/residents problem with ties*, in Proceedings of the 14^{th} International Conference on Computing and Communication Technologies (RIVF 2020), RMIT, Ho Chi Minh, Apr.6-7, 2020,pp.1- 6, Lecture Notes in Computer Science (**SCOPUS**), Springer, ISBN 978-1-7281-5377-3.

[A.4] **Nguyen Thi Uyen**, Nguyen Long Giang, Tran Xuan Sang and Hoang Huu Viet "*An efficient heuristics algorithm for solving the Student- Project Allocation with Preferences over Projects*", 24^{th} Hội thảo Quốc gia (VNICT 2021), Thai Nguyen, Việt Nam, Dec. 13-14, pp. 1-6, 2021.

[A.5] **Nguyen Thi Uyen**, Giang L. Nguyen, Canh V. Pham, Tran Xuan Sang and Hoang Huu Viet: "*A Heuristic Algorithm for the Student-Project Allocation Problem with Lecturer Preferences over Students with Ties*, in Proceedings of the 11^{th} International Conference on Computational Data and Social Networks (CSoNET 2022), Tampa, Florida, USA, Dec. 5-7, 2022, in Press, Lecture Notes in Computer Science (**SCOPUS**), Springer.

[B.1] **Nguyen Thi Uyen**, Giang L. Nguyen and Hoang Huu Viet, "*An effi- cient Heuristic search algorithm for the Hospitals/Residents with Ties problem*", Applied Artificial Intelligence (đang gửi tạp chí).

[B.2] **Nguyen Thi Uyen**, Nguyen Long Giang and Hoang Huu Viet "*Faster and Simpler Heuristic Algorithm for the Student-Project Allocation with Preferences over Projects*", International Journal of Fuzzy Logic and Intelligent Systems (đang gửi tạp chí).

2. Other publications

[C.1] **Nguyen Thi Uyen** and Tran Xuan Sang, "*An efficient algorithm to find a maximum weakly stable matching for SPA-ST problem*, in Proceed- ings of the 21^{st} International Conference on Artificial Intelligence and Soft Computing (ICAISC 2022), Zakopane, Poland, Jun. 18-22, 2022, in Press, Lecture Notes in Artificial Intelligence (**SCOPUS**), Springer.

[C.2] Hoang Huu Viet, **Nguyen Thi Uyen**, Cao Thanh Sơn, and Le Hong Trang, "*Một thuật toán tìm kiếm cục bộ giải bài toán phân công địa điểm thực tập cho sinh viên*", 23^{th} Hội thảo Quốc gia (VNICT 2020), Hạ Long, Việt Nam, Nov. 5-6, pp. 271–276, 2020.

INTRODUCTION

1. The urgency of the thesis

The Stable Marriage Problem (SMP) is a well-known matching problem first introduced by Gale and Shapley in 1962. An instance of SMP size n, denoted by I, consists of a set of n men and women. Each person has a rank list, in which each person ranks the opposite sex in a certain order of preference. The goal of the problem is to find matching between men and women that satisfies stability according to some criteria. Recently, the SMP problem has received much attention from researchers in the fields of Artificial Intelligence and Optimal Computing.

- *In terms of practice*: In 2012, Shapley and Roth were awarded the Nobel Prize in Economics for their achievements based on models derived from the SMP problem in the field of stock market management in the United States. Besides, 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 problem, the Student-Project Allocation problem, the Stable Marriage Roommates problem and the problem of optimizing service requests of Internet users to telecommunications carriers. Therefore, it is necessary to study the stable marriage problem and its variants to find the optimal solutions to the matching problem in practical applications.

-*In terms of science*: Several variants of the SMP have been proposed recently, such as the Stable Marriage problem with Ties (SMT), the Stable Marriage problem with Incomplete (SMI), the Stable Marriage Problem with Ties and Incomplete lists (SMTI). According to, with the appearance of ties in the preference lists, Irving et al. (2002) has shown that there are three stability criteria of a matching are defined, including *weakly stable*, *strongly stable* and *super-stable*. The authors proved that a weakly stable matching always exists, while strong and superstable matching may not exist for all instances of SMT and SMTI problems. In addition, the authors have also demonstrated, finding a weakly stable matching with maximum size (MAX) is an NP-hard problem. In this study, the thesis focused on finding a weakly stable matching with the maximum size, so for simplicity, the thesis calls the weakly stable matching a stable matching. To study this problem, it is necessary to research heuristic algorithms to find a weakly stable coupling with maximum size for MAX-SMTI and variants.

2. Research objectives of the thesis

- Research overview of the stable marriage problem and variants.

- Research and propose heuristic algorithms to solve problems MAX-SMTI, MAX-HRT, and MAX-SPA.

3. The main contents of the thesis.

Chapter 1. Overview. In this chapter, the thesis presents an overview of the theoretical basis and research situation of the stable marriage problem and its variations.

Chapter 2. Proposed algorithms for solving the MAX-SMTI problem. The thesis proposes 02 algorithms to solve the MAX-SMTI problem in this chapter. The results are published in the specialized journal SCIE and international conferences with the SCOPUS index.

Chapter 3. Proposed algorithms for solving the MAX-HRT problem. The thesis proposes 02 algorithms to solve the MAX-HRT problem in this chapter. The results are published in the specialized international conferences with the SCOPUS index.

Chapter 4. Proposed algorithms for solving the MAX-SPA problem. The thesis proposes 02 algorithms to solve the MAX-SPA problem in this chapter. The results are published in the specialized international conferences with the SCOPUS index.

CHAPTER 1. OVERVIEW OF STABLE MARRIAGE PROBLEM

1.1. The Stable Marriage problem

An instance of SMP size n, denoted by I, consists of a set of n men and women. Each person has a rank list, in which each person ranks the opposite sex in a certain order of preference. An example of an instance including 8 men and 8 women is shown in Table 1.1. Gale and Shapley (1962) showed

Table 1.1: *An instance of* SMP

m_i's rank list, $m_i \in \mathcal{M}$	w_j's rank list, $w_j \in \mathcal{W}$
m_1: $w_4\ w_3\ w_1\ w_5\ w_2\ w_6\ w_8\ w_7$	w_1: $m_4\ m_7\ m_3\ m_8\ m_1\ m_5\ m_2\ m_6$
m_2: $w_2\ w_8\ w_4\ w_5\ w_3\ w_7\ w_1\ w_6$	w_2: $m_5\ m_3\ m_4\ m_2\ m_1\ m_8\ m_6\ m_7$
m_3: $w_5\ w_8\ w_1\ w_4\ w_2\ w_3\ w_6\ w_7$	w_3: $m_2\ m_8\ m_6\ m_4\ m_3\ m_7\ m_5\ m_1$
m_4: $w_6\ w_4\ w_3\ w_2\ w_5\ w_8\ w_1\ w_7$	w_4: $m_5\ m_6\ m_8\ m_3\ m_4\ m_7\ m_1\ m_2$
m_5: $w_6\ w_5\ w_4\ w_8\ w_1\ w_7\ w_2\ w_3$	w_5: $m_1\ m_8\ m_5\ m_2\ m_3\ m_6\ m_4\ m_7$
m_6: $w_7\ w_4\ w_2\ w_5\ w_6\ w_8\ w_1\ w_3$	w_6: $m_8\ m_6\ m_2\ m_5\ m_1\ m_7\ m_4\ m_3$
m_7: $w_8\ w_5\ w_6\ w_3\ w_7\ w_2\ w_1\ w_4$	w_7: $m_5\ m_2\ m_8\ m_3\ m_6\ m_4\ m_7\ m_1$
m_8: $w_4\ w_7\ w_1\ w_3\ w_5\ w_8\ w_2\ w_6$	w_8: $m_4\ m_5\ m_7\ m_1\ m_6\ m_2\ m_8\ m_3$

that there exists at least one stable matching for every instance of SM and proposed an algorithm, called the Gale-Shapley algorithm, to find a stable matching of SM instances of size n in time $O(n^2)$. In addition, some other research methods have also been proposed such as approximation algorithms, heuristic algorithms, and other research methods. However, the SMP has little application in practice because of the strict constraints of the favorite list, i.e., each man must rank all the women and vice versa. Therefore, some variants of the SMP problem have been introduced and applied in practice in recent years.

1.2. Variation of the Stable Marriage problem

The first is Stable Marriage Problem with Ties (SMT), meaning that each person can rank two or more people of the opposite sex in an equal order on the rank list. The second is the Stable Marriage Problem with Incomplete (SMI), meaning that each person only ranks some people of the opposite sex on their rank list. If we combine the two variants SMT and SMI, we have a Stable marriage problem with Ties and Incomplete lists (SMTI). The goal of the SMTI problem is to find a matching that is not only stable but also has the maximum number of matched men, also known as the MAX-SMTI. Manlove et al (2008) demonstrated that MAX-SMTI is an NP-hard problem, therefore, finding an efficient algorithm to solve the problem of large sizes is a challenge for researchers. In this thesis, we focus on the SMTI and its variants.

Definition 1.1 (SMTI instance)**.** *An* SMTI *instance of size* n *involves a set* $\mathcal{M} = \{m_1, m_2, \cdots, m_n\}$ *of men and a set* $\mathcal{W} = \{w_1, w_2, \cdots, w_n\}$ *of women in which each person ranks some members of the opposite sex in order of preference, i.e., the rank list of each person may include ties and be incomplete.*

We denote $rank(m_i, w_j)$ such that rank of $w_j \in \mathcal{W}$ in m_i's rank list and $rank(w_j, m_i)$ is rank of $m_i \in \mathcal{M}$ ($m_i \in \mathcal{M}$) in w_j's rank list ($w_j \in \mathcal{W}$). If $m_i \in \mathcal{M}$ prefers $w_j \in \mathcal{W}$ to $w_k \in \mathcal{W}$, meaning that $rank(m_i, w_j) < rank(m_i, w_k)$ and if $m_i \in \mathcal{M}$ prefers $w_j \in \mathcal{W}$ and $w_k \in \mathcal{W}$ the same ranks $rank(m_i, w_j) = rank(m_i, w_k)$.

Definition 1.2 (Acceptable pair)**.** *A pair* (m_i, w_j) *is called an acceptable pair, if* $rank(m_i, w_j) > 0$ *and* $rank(w_j, m_i) > 0$.

Definition 1.3 (Matching)**.** *A matching* M *of* SMTI *instance is a set of acceptable pairs such that each person belongs to at most one pair* $M = \{(m_i, w_j) \in \mathcal{M} \times \mathcal{W} | rank(m_i, w_j) > 0$ *and* $rank(w_j, m_i) > 0\}$*, where each* $m_i \in \mathcal{M}$ *only assigns to* $w_j \in \mathcal{W}$ *and vice versa. If* $(m_i, w_j) \in M$*, then* m_i *and* w_j *are partners, denoted by* $M(m_i) = w_j$ *and* $M(w_j) = m_i$*. If* $m_i \in \mathcal{M}$ *is an unassigned to* M*, then* m_i *is called single and denoted by* $M(m_i) = \varnothing$*. Similarly, if* $w_j \in \mathcal{W}$ *is an unassigned in* M*, then* w_j *is called single and denoted by* $M(w_j) = \varnothing$.

Definition 1.4 (Blocking pair)**.** *A* $(m_i, w_j) \in \mathcal{M} \times \mathcal{W}$ *is blocking pair in* M *if:*

1. $rank(m_i, w_j) > 0$ *and* $rank(w_j, m_i) > 0$*;*
2. $M(m_i) = \varnothing$ *or* $rank(m_i, w_j) < rank(m_i, M(m_i))$*;*
3. $M(w_j) = \varnothing$ *or* $rank(w_j, m_i) < rank(w_j, M(w_j))$.

Definition 1.5 (Dominated blocking pair)**.** *A blocking pair* (m_i, w_j) *dominates a blocking pair* (m_i, w_k) *if* $rank(m_i, w_j)$¡$rank(m_i, w_k)$.

Definition 1.6 (Undominated blocking pair)**.** *A blocking pair* (m_i, w_j) *is undominated if there are no other blocking pairs dominating* (m_i, w_k) *such that* $rank(m_i, w_k) < rank(m_i, w_j)$.

Definition 1.7 (Stable matching). *A matching M is called stable if it admits no blocking pair, otherwise, it is called unstable.*

Definition 1.8 (Matching size). *Matching size is the number of assigned men in a stable matching M, denoted by $|M|$.*

Definition 1.9 (Perfect matching). *A matching M is called perfect if $|M| = n$, otherwise, it is called non-perfect.*

Table 1.2: *An instance of* SMTI

m_i's rank list, $m_i \in \mathcal{M}$	w_j's rank list, $w_j \in \mathcal{W}$
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

Several research directions have been proposed to solve MAX-SMTI problem such as:

i) Approximation algorithms: There are several approximation algorithms proposed to consider lower bounds for the MAX-SMTI problem. In general, the problem MAX-SMTI has been solved quite well with relatively good solution quality and with an increasing approximate ratio compared to the previously proposed algorithms. The best approximation algorithm is $3/2$. Although the approximation algorithms have solved the problem relatively well in terms of time and quality of solutions, we found that it is possible to improve the quality of solutions better. In addition, these algorithms do not show effective experiments to solve the SMTI problem with a large size.

ii) Heuristics algorithms: 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 on solution quality. Recently, local search approaches to deal with the MAX-SMTI problem have been applied by some researchers. Gelain et al (2013) proposed a local search algorithm, namely

LTIU for MAX-SMTI. Munera et al (2015) modeled SMTI s a permutation problem and applied the adaptive search method, called AS to solve the problem.

iii) Other approaches: In addition, some researchers have proposed new approaches to solving the SMTI problem such as Integer Programming (IP), SAT model, and Integer Linear Programming (ILP).

1.3. Variants of the SMTI

1.3.1. The Hospitals/Residents with Ties

The Hospitals/Residents with Ties problem (HRT), is a variant of SMTI problem. An instance I of HRT consists of a set $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$ of *residents* and a set $\mathcal{H} = \{h_1, h_2, \cdots, h_m\}$ of *hospitals* in which each resident $r_i \in \mathcal{R}$ ranks in strict order a subset of $\mathcal{H}$ in its *rank list*, each hospital $h_j \in \mathcal{H}$ ranks in strict order applicants in its rank list, and each hospital $h_j \in \mathcal{H}$ has a *capacity* $c_j \in \mathbb{Z}^+$ indicating the maximum number of residents that can be assigned to it.

Table 1.3 shows an instance of HRT of 8 residents and 5 hospitals.

Table 1.3: *An instance of* HRT

r_i's rank list, $r_i \in \mathcal{R}$	h_j's rank list, $h_j \in \mathcal{H}$
r_1: $h_1\ h_3\ h_2$	h_1: $r_3\ (r_7\ r_5\ r_2)\ r_4\ r_6\ r_1$
r_2: $h_1\ (h_5\ h_4)\ h_3$	h_2: $r_5\ r_6\ (r_3\ r_4)\ r_1$
r_3: $h_1\ h_5\ h_2$	h_3: $(r_5\ r_2)\ r_6\ r_1\ r_7$
r_4: $h_1\ (h_2\ h_4)$	h_4: $r_8\ r_2\ r_4\ r_7$
r_5: $h_3\ h_1\ h_2$	h_5: $r_3\ (r_7\ r_6\ r_8)\ r_2$
r_6: $(h_3\ h_2)\ h_1\ h_5$	
r_7: $h_3\ h_4\ h_5\ h_1$	
r_8: $h_5\ h_4$	
h_j's capacity, $h_j \in \mathcal{H}$: $c_1 = 2$, $c_2 = 3$, $c_3 = c_4 = c_5 = 1$	

1.3.2. The Student-Project Allocation

In project-based courses, students have to be assigned to projects offered by lecturers. The question for this problem is how to allocate students to projects to meet the requirements of students and lecturers. To solve this problem, Abraham et al (2003) introduced a formal definition of the *Student-Project Allocation problem* (SPA). Recently, variants of SPA-P and SPA-ST have been of great interest to the research community in practical applications. Therefore, the thesis focuses on studying two variants of the SPA problem, which is the problem of SPA-P và SPA-ST. The goal of the problem is to find a stable matching with the maximum size, that is, the maximum number of students who receive the projects satisfy the constraints on the capacity of the lecturers and the projects (MAX-SPA). Given SPA-P instance in Table 1.4 and SPA-ST instance in Table 1.5.

Table 1.4: *An instance of* SPA-P

s_i's rank list, $s_i \in \mathcal{S}$	l_k's rank list, $l_k \in \mathcal{L}$
s_1: p_1 p_3 p_4	l_1: p_1 p_2 p_3
s_2: p_5 p_1	l_2: p_4 p_5
s_3: p_2 p_5	
s_4: p_4 p_2	
s_5: p_5	
p_j's capacity $p_j \in \mathcal{P}$: $c_1 = c_2 = c_3 = c_4 = 1, c_5 = 2$	
l_k's capacity, $l_k \in \mathcal{L}$: $d_1 = 3$, $d_2 = 2$	

Table 1.5: *An instance of* SPA-ST

s_i's rank list, $s_i \in \mathcal{S}$	l_k's rank list, $l_k \in \mathcal{L}$
s_1: $(p_1\ p_7)$	l_1: $(s_7\ s_4)$ s_1 s_3 $(s_2\ s_5)$ s_6
s_2: p_1 p_3 p_5	l_2: s_3 s_2 s_7 s_5
s_3: $(p_2\ p_1)$ p_4	l_3: $(s_1\ s_7)$ s_6
s_4: p_2	
s_5: p_1 p_4	l_1 offers p_1, p_2, p_3
s_6: p_2 p_8	l_2 offers p_4, p_5, p_6
s_7: $(p_5\ p_3)$ p_8	l_3 offers p_7, p_8
p_j's capacity, $p_j \in \mathcal{P}$: $c_1 = 2, c_j = 1, (2 \leq j \leq 8)$	
l_k's capacity, $l_k \in \mathcal{L}$: $d_1 = 3, d_2 = 2, d_3 = 2$	

1.3.3. Related works

Several approximation algorithms have been proposed to solve the MAX-HRT problem with ratio approximation $3/2$. Munera et al.(2015) modeled Adaptive Search and Gelain et al. (2013) proposed a Local search algorithm for solving HRT problem. However, the proposed algorithms have not effectively solved the problem regarding solution quality and execution time for HRT large size. Therefore, this thesis focuses on researching and proposing heuristics algorithms to solve the problem MAX-HRT with large size. For the SPA-P problem, where a stable join can have different sizes, finding a join that is both stable and has a maximum size is an NP-hard problem. Cooper and Manlove (2018) proposed a 3/2 approximation algorithm to find a stable matching with maximum size for the SPA-ST problem.

CHAPTER 2. PROPOSED ALGORITHMS TO SOLVE MAX-SMTI PROBLEM

2.1. Max-Conflicts Algorithm

This section proposes a Max-Conflicts-based heuristic search, called MCS shown in Algorithm 2.1, to solve the MAX-SMTI problem.

Algorithm 2.1: MCS algorithm

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

Output: A matching M.

```
function Main(I)
    Randomly generate M;
    M_best := M;
    f_best := n;
    iter := 0;
    while (iter ≤ max_iters) do
        X := Find_UBPs(M);
        if (X = ∅) then
            if (f_best > f(M)) then
                M_best := M;
                f_best := f(M);
            if (f_best > 0) then
                M := Escape_Local_Minima(M);
                continue;
            else
                break;
        for (each m_i ∈ X) do
            ubp(w_j) := ubp(w_j) + 1, where (m_i, w_j) ∈ X;
        for (each m_i ∈ X) do
            h(m_i) := n * ubp(w_j) − rank(w_j, m_i);
        if (a small p) then m_j := a random m_i ∈ X;
        else m_j := argmax(h(m_i)), ∀m_i ∈ X ;
        Remove blocking pair (m_j, w_k);
        iter := iter + 1;
    return M_best;
end function
```

The main idea of MCS is that starting from a random matching, the algorithm will find a set of UBP and define a heuristic function to select the best UBP and remove them from the matching. Let $X = \{(m_i, w_j) \mid m_i \in \mathcal{M}, w_j \in \mathcal{W}\}$ denote a set of UBP 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 UBP formed by woman $w_j \in X$, we define a heuristic function as follows:

$$h(m_i) = n \times ubp(w_j) - rank(w_j, m_i), \forall (m_i, w_j) \in X. \qquad (2.1)$$

Algorithm 2.2: Finding a set UBPs, X

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

```
function Find_UBPs(M)
    X := ∅;
    for ( each m_i ∈ 𝓜) do
        w_j := M(m_i);
        while (∃w_k ∈ 𝓦|rank(m_i, w_k) > 0) do
            w_k := argmin(rank(m_i, w_k) > 0);
            if (rank(m_i, w_k) = rank(m_i, w_j)) then
                break;
            if ((m_i, w_k) is a blocking pair) then
                X := X ∪ (m_i, w_k);
                break;
            else
                Delete w_k in m_i's rank list;
    return X;
end function
```

First, MCS finds a set X of UBP pairs for M using Algorithm 2.2. Second, MCS checks if X is empty, i.e., M is a stable matching, and if *(i)* M_{best} has a number of singles more than M, then M is assigned to M_{best}; *(ii)* M_{best} is a non-perfect matching, i.e., the MCS is stuck at the local minimum, then the MCS calls Algorithm 2.3 to overcome the local minimum and performs the next iteration, otherwise, MCS returns perfect matching M_{best}. Third, MCS counts the number of UBP, $ubp(w_j)$, generated by each $w_j \in X$ and determines the heuristic values, $h(m_i)$, for each $m_i \in X$. Fourth, MCS ran-

Algorithm 2.3: Escape from local minima

Input: A matching M.
Output: A matching M.

function `Escape_Local_Minima` (M)
  if *(a probability* $p \leq 0.5$*)* **then**
    $U := \{m_i \mid M(m_i) = \varnothing\}$;
    Randomly generate $m_j \in U$;
    for (*each* $w_k \in m_j$*'s rank list*) **do**
      if $(M(w_k) \neq \varnothing)$ **then**
        break pair $(M(w_k), w_k)$ into two singles, $M(w_k)$ and w_k;
  else
    $V := \{w_i \mid M(w_i) = \varnothing\}$;
    Randomly generate $w_j \in V$;
    for (*each* $m_k \in w_j$*'s rank list*) **do**
      if $(M(m_k) \neq \varnothing)$ **then**
        break pair $(m_k, M(m_k))$ into two singles, m_k and $M(m_k)$;
  return M;
end function

domly chooses a $m_j \in X$ with small probability p or chooses a $m_j \in X$ corresponding to the maximum $h(m_j)$. The MCS repeats until a perfect matching is found or the number of iterations has reached the maximum.

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. Experiments showed that MCS outperforms LTIU and AS algorithms in terms of execution time and solution quality for the MAX-SMTI problem. The experimental results have shown that MCS is more efficient than algorithms LTIU and AS in terms of the quality of the solution and the execution time for the MAX-SMTI problem.

2.2. Heuristic-Repair Algorithm

The proposed HR algorithm includes the algorithm GS to find a stable concatenation and a heuristic function to improve the size of the matching found by GS for an instance of SMTI is described in Algorithm 2.4. First, HR finds a stable matching by improving the idea of GS.

Algorithm 2.4: HR algorithm

Input: - An instance I of SMTI.
- The maximum number of iterations, max_iters.

Output: A matching M.

function `Main(`I`)`
  for $(each\ m_i \in M)$ **do**
    $M(m_i) := \varnothing$;
    $a(m_i) := 1$;
    $c(m_i) := 0$;
  $iter := 1$;
  while $iter \leq max\_iters$ **do**
    m_i := some man is active, i.e., $a(m_i) = 1$;
    if $\nexists a(m_i) = 1$ **then**
      if $|M| = n$ **then** break;
      $iter := iter + 1$;
      $M :=$ `Improve(`M`)`;
      continue;
    if *m_i's rank list is empty* **then**
      $a(m_i) := 0$;
      $c(m_i) := c(m_i) + 1$;
      continue;
    if *there exists a single woman w_j to whom m_i prefers most* **then**
      $M(m_i) := w_j$;
      $a(m_i) := 0$;
    else
      w_j := a woman to whom m_i prefers most;
      $m_k := M(w_j)$;
      if *there exists a single w_t that $rank(m_k, w_t) = rank(m_k, w_j)$* **then**
        `repair(`m_i, m_k`)`;
      if *$M(m_i) = \varnothing$ and $rank(w_j, m_i) < rank(w_j, m_k)$* **then**
        `repair(`m_i, m_k`)`;
        $rank(m_k, w_j) := 0$;
      else
        $rank(m_i, w_j) := 0$;
  return M;
end function

Algorithm 2.5: Improve a stable matching M

Input: A stable matching M.

Output: A stable matching M.

```
function Improve(M)
    for each single man m_i ∈ M do
        recover m_i's original rank list;
        X := {};
        for each w_j ∈ m_i's rank list do
            m_k := M(w_j);
            if rank(m_i, w_j) ≤ rank(m_k, w_j) or
              rank(w_j, m_i) = rank(w_j, m_k) then
                X := X ∪ {w_j};
        if X = ∅ then continue;
        for each w_j ∈ X do
            m_k := M(w_j);
            k := number of w_t in m_k's rank list, where
              rank(m_k, w_t) = rank(m_k, w_j);
            h(w_j) := 1/k+(rank(w,m_i)−rank(w_j, m_k))×(1−c(m_k));
        w_j := argmin(h(w_j)), ∀w_j ∈ X;
        repair(m_i, m_k), where m_k := M(w_j);
        rank(m_k, w_j) := 0;
    return M;
end function
```

Then, the HR algorithm re-applies the improved GS algorithm. If a stable matching is found that has not reached its maximum size, HR calls Algorithm 2.5 to improve the size of the M by proposing a heuristic function. The HR algorithm terminates when a stable matching with maximum size or the maximum number of iterations is reached. 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 experimental results for large randomly generated instances of SMTI showed that HR outperforms GSA2 and MCS in terms of solution quality for finding perfect matchings of MAX-SMTI problem.

CHAPTER 3. PROPOSED ALGORITHM TO SOLVE MAX-HRT PROBLEM

3.1. Min-Conflicts Algorithm

This section presents a Min-Conflicts algorithm, named MCA to solve MAX-HRT problem in Alg. 3.1.

Algorithm 3.1: Min-Conflicts Algorithm

Input: - An instance I of HRT.
- A small probability p.
- The maximum iterations, max_iters.
Output: A matching M.

function `MCA(`I`)`
  Randomly generate M;
  $M_{best} := M$;
  $f_{best} := n$;
  $iter := 0$;
  while $(iter \leq max\_iters)$ **do**
    $iter := iter + 1$;
    $[f(M), X] :=$ `Find_Cost_And_UBPs(`M`)`;
    if $(X = \emptyset)$ **then**
      if $(f_{best} > f(M))$ **then**
        $M_{best} := M$;
        $f_{best} := f(M)$;
      if $(f_{best} > 0)$ **then**
        $M :=$ a randomly generated matching;
        continue;
      else
        break;
    if *(a small probability of p)* **then**
      $r_j :=$ a random resident $r_i \in X$;
    else
      $r_j := \text{argmin}(rank(h_k, r_i)), \forall (r_i, h_k) \in X$;
    Remove blocking pair $(r_j, X(r_j))$;
  return M_{best};
end function

Initially, the algorithm generates a random matching M and assigns the best matching M_{best} to M. At each iteration, MCA finds the set of UBP, $X = \{(r_i, h_j) \in R \times H\}$, for M. Then, the algorithm calls the function in Alg. 3.2 to find the cost, $f(M) = \#nbp(M) + \#nur(M)$, where $\#nbp(M)$ is the number of UBP for M and $\#nur(M)$ is the number of unassigned residents in M. If M is not perfect, the algorithm restarts a new matching M and continues the next iteration. Then, the algorithm checks if a small probability of p is accepted, it chooses a random resident $r_j \in X$. Otherwise, it chooses the resident, $r_j \in X$, such that it is most preferred by hospital $h_k \in X$. The algorithm repeats until either M_{best} is a perfect matching or a maximum number of iterations is reached. In the latter case, the algorithm returns either a maximum stable matching or an unstable matching.

Algorithm 3.2: Find the cost and UBPs of a matching

Input: A matching M.
Output: The cost, $f(M)$, and the set of UBPs, X.

```
function Find_Cost_And_UBPs(M)
    X := ∅;
    #nur := 0;
    #nbp := 0;
    for (each r_i ∈ R) do
        ubp := false;
        while (there exists a h_j ∈ H in r_i's rank list) do
            h_j := a h_j such that r_i prefers most;
            if (rank(r_i, h_j) = rank(r_i, M(r_i))) then
                break;
            if ((r_i, h_j) is a blocking pair) then
                X := X ∪ (r_i, h_j);
                #nbp := #nbp + 1;
                ubp := true;
                break;
            else
                Delete h_j in r_i's rank list;
        if ((ubp = false) and (r_i is unassigned)) then
            #nur := #nur + 1;
    f(M) := #nbp + #nur;
    return (f(M), X);
end function
```

We present experimental results to compare the execution time and solution quality of MCA with those of LTIU. All the experiments were implemented by Matlab software on a personal computer with a Core i7-8550U CPU 1.8GHz and 16 GB memory. Experiments showed that MCA is efficient in terms of execution time and solution quality for MAX-HRT.

3.2. Heuristic-Search Algorithm

This section proposes a Heuristic Search algorithm, named HS to find a maximum stable matching for MAX-SMTI shown in Algorithm 3.3. Starting from a random matching M, at each iteration, if there exists a r_i such that $a(r_i) = 1$, HS finds a h_j and a list wait $W(r_i)$ shown in Algorithm 3.4 such that (r_i, h_j) is an UBP and $W(r_i)$ is a set of $h_k \in \mathcal{H}$ where $rank(r_i, h_k) < rank(r_i, M(r_i))$ or $rank(r_i, h_k) = rank(r_i, h_j)$. If there no exists h_j, such that (r_i, h_j) is a blocking pair, the algorithm assigns $a(r_i) = 0$. If there are no active residents, HS means that HS finds a stable matching M. If the algorithm finds a non-perfect matching and we consider this is the case of getting stuck at a local minimum. Therefore, the algorithm calls the function given in Algorithm 3.5 to overcome the local minimum and continues the next iteration. The algorithm terminates when it finds a perfect matching or reaches a given maximum number of iterations. In the latter case, the algorithm returns a stable matching of the maximum size found so far.

We implemented all the HS, AS, and HP algorithms in Matlab 2019a software and ran them on a computer with Core i7-8550U CPU1.8 GHz and 16 GB RAM in Windows 10. The experimental results show that our algorithm not only finds a much higher percentage of perfect matchings but also runs much faster than AS and HP algorithms for MAX-HRT problem.

Algorithm 3.3: HS Algorithm

```
Input: - An instance $I$ of HRT.
       - A maximum number $max\_iter$.
Output: A stable matching $M$.
function HS ($I$)
    Randomly generate $M$;
    $M_{best} := M$;
    $a(r_i) := 1, \forall r_i \in \mathcal{R}$;
    $y(r_i) := 0, \forall r_i \in \mathcal{R}$;
    $z(h_j) := 0, \forall h_j \in \mathcal{H}$;
    $rank^{\dagger}(r_i, h_j) := rank(r_i, h_j), \forall r_i \in \mathcal{R}, h_j \in \mathcal{H}$;
    $iter := 0$;
    while $iter \leq max\_iter$ do
        $iter := iter + 1$;
        $r_i :=$ an active $r_i$;
        if ($\nexists r_i$ such that $a(r_i)$ = 1) then
            if $|M_{best}| < |M|$ then
                $M_{best} := M$;
                if ($|M_{best}| = n$) then break;
            $M' :=$ Improve $M$;
            if $M' = M$ then break;
            $M := M'$;
            continue;
        $[h_j, W(r_i)] :=$ Find_UBP_and_Wait_List $r_i, M$;
        if $h_k \neq \varnothing$ then
            $M := M \setminus \{(r_i, h_k)\} \cup \{(r_i, h_j)\}$, where $h_k := M(r_i)$;
            $a(r_i) := 0$;
            for $r_l$ such that $W(r_l) = h_k$ do
                $rank(r_l, h_k) := rank^{\dagger}(r_l, h_k)$;
                $a(r_l) := 1$;
            if $|M(h_j)| > c_j$ then
                $r_w := h_j$'s worst resident;
                $M := M \setminus \{(r_w, h_j)\}$;
                $a(r_w) := 1$;
        else
            $a(r_i) := 0$;
    return $M_{best}$;
end function
```

Algorithm 3.4: Finding an UBP and $W(r_i)$

Input: - A resident r_i and matching M.
Output: - A h_j such that (r_i, h_j) is an UBP.
- A wait list $W(r_i)$.

function `Find_UBP_and_Wait_List` (r_i, M)
    for $h_k \in r_i$*'s rank list* **do**
        $f(h_k) := rank(r_i, h_k) + |M(h_k)|/(c_k + 1)$;
    $h_j := \varnothing$;
    while r_i*'s rank list is not empty* **do**
        $h_k := \text{argmin}(f(h_k) \geq 1), \forall h_k \in \mathcal{H}$;
        if $rank(r_i, h_k) = rank(r_i, M(r_i))$ **then**
            break;
        if (r_i, h_k) *is a blocking pair* **then**
            $h_j := h_k$;
            break;
        else
            $W(r_i) := W(r_i) \cup h_k$;
            $rank(r_i, h_k) := 0$;
            $f(h_k) := 0$;
    return h_j và $W(r_i)$;
end function

Algorithm 3.5: Improve stable matching

Input: A stable matching, M.
Output: A matching, M.

function `Improve_Matching`(M)
  for *each* $r_u \in \mathcal{R}$, *such that* $M(r_u) = \varnothing$ **do**
    Find $(r_i, h_k) \in M$ such that $rank(h_k, r_i) = rank(h_k, r_u)$;
    if $y(r_u) \geq y(r_i)$ **then**
      $M := M \setminus \{(r_i, h_k)\} \cup \{(r_u, h_k)\}$;
      $y(r_u) := y(r_u) + 1$;
      $a(r_u) := 0$;
      $a(r_i) := 1$;
      $rank(r_u, h_k) := rank^{\dagger}(r_u, h_k)$;
  for *each* $h_t \in \mathcal{H}$, *such that* $|M(h_t)| < c_j$ **do**
    Find $(r_i, h_k) \in M$ such that $rank^{\dagger}(r_i, h_k) = rank^{\dagger}(r_i, h_t)$
    if $z(h_t) \geq z(h_k)$ **then**
      $M := M \setminus \{(r_i, h_k)\} \cup \{(r_i, h_t)\}$;
      $z(h_t) := z(h_t) + 1$;
      $rank(r_i, h_t) := rank^{\dagger}(r_i, h_t)$;
    for *each* r_w *such that* $W(r_w) = h_k$ **do**
      $rank(r_w, h_k) := rank^{\dagger}(r_w, h_k)$;
      $a(r_w) := 1$;
  return M;
end function

CHAPTER 4. PROPOSED ALGORITHMS TO SOLVE MAX-SPA PROBLEM

4.1. SPA-P-heuristic algorithm for MAX-SPA-P problem

Algorithm 4.1: SPA-P-heuristic Algorithm

Input: An instance I ofSPA-P.
Output: A stable matching M.

```
function SPA-P-heuristic(I)
    M := ∅;
    a(s_i) := 1, ∀s_i ∈ S;
    h_{l_k}(s_i) := 0, ∀l_k ∈ L, ∀s_i ∈ S;
    while there exists an active s_i do
        if s_i's rank list is empty then
            a(s_i) := 0;
            continue;
        p_j := the most prefered project in s_i's rank list;
        l_k := lecturer who offers p_j;
        M := M ∪ {(s_i, p_j)};
        a(s_i) := 0;
        y(s_i) := number of project ranked by s_i;
        h_{l_k}(s_i) := rank(l_k, p_j) + y(s_i)/(q + 1);
        if |M(p_j)| > c_j then
            s_t := argmax(h_{l_k}(s_t)), ∀s_t ∈ M(p_j);
            M := M \ {(s_t, p_j)};
            rank(s_t, p_j) := 0;
            h_{l_k}(s_t) := 0;
            a(s_t) := 1;
        if |M(l_k)| > d_k then
            s_t := argmax(h_{l_k}(s_t)), ∀s_t ∈ M(l_k);
            p_z := M(s_t);
            M := M \ {(s_t, p_z)};
            rank(s_t, p_z) := 0;
            h_{l_k}(s_t) := 0;
            a(s_t) := 1;
    return M;
end function
```

This section presents heuristic algorithm, called SPA-P-heuristic for MAX-SPA-P problem, shown in Algorithm 4.1. Initialize from an empty matching M. At each iteration, when a student s_i is assigned to the most preferred project p_j in her/his list to form a pair $(s_i, p_j) \in M$ if the project p_j or the lecturer l_k who offers p_j is over-subscribed, then an arbitrary student s_r in $M(p_z)$, where p_z is l_k's worst non-empty project, is removed from M. We recognize that if three following conditions are met: (*i*) $M(p_z)$ consists of at least two students s_r and s_t; (*ii*) s_r ranks only one project; and (*iii*) s_t ranks more than one project, then if we remove s_r from M, then s_r is unassigned in M forever, while if we remove s_t from M, then s_t can be assigned to some project in her/his list at the next iterations. To solve this problem, we propose a heuristic function as follows:

$$h_{l_k}(s_t) = rank(l_k, p_z) + y(s_t)/(q + 1). \tag{4.1}$$

where l_k is a lecturer who offers project p_z and $y(s_t)$ is the number of projects ranked by s_t. If p_j is *over-subscribed*, then the worst student s_t in $M(p_j)$ is removed from M. If so, s_t deletes p_j in her/his list and she/he becomes active again. If l_k is *over-subscribed*, then the worst student s_t in $M(l_k)$ is removed from M. If so, s_t deletes p_z in her/his list, where p_z is assigned to s_t, and she/he becomes active again. When a student s_t is removed from M, the heuristic value $h_{l_k}(s_t)$ is assigned to zero. The algorithm is repeated until all students are inactive and it returns a maximum stable matching.

Our experimental results show that our proposed algorithm outperforms SPA-P-approx, SPA-P-promotion, and SPA-P-MCH algorithms in terms of solution quality and execution time for MAX-SPA-P problem.

4.2. HAG algorithm for MAX-SPA-ST problem

This section proposes a heuristic algorithm, called HAG shown in Aglorithm 4.2 for MAX-SPA-ST problem of large size. The algorithm starts from an empty matching, $\mathcal{M} = \varnothing$. At each iteration, HAG considers an unassigned student $s_i \in \mathcal{S}$ whose ranking list of s_i is not empty and determines the heuristic function $h(p_j)$ for each project p_j in s_i's rank list, where p_j is offered by l_k to choose the best project based on the minimum value of $h(p_j)$ as follows:

$$h(p_j) = rank(s_i, p_j) - min(d_k - |M(l_k)|, 1)/2 - (c_j - |M(p_j)|)/(2 \times c_j + 1). \tag{4.2}$$

Algorithm 4.2: HAG algorithm for MAX-SPA-ST

Input: An SPA-ST instance I

Output: A stable matching M.

```
function HAG (I)
    M := ∅
    v(s_i) := 0, ∀s_i ∈ S
    while true do
        s_i := an unassigned student that s_i's rank list is non-empty
        if there exists no student s_i then
            if |M| = n then break
            else
                M' := Escape(M)
                if M' = M then break
                M := M'
                continue
        for each p_j ∈ A_i do
            l_k := a lecturer who offers p_j
            h(p_j) = rank(s_i, p_j)-min(d_k-|M(l_k)|, 1)/2-(c_j-|M(p_j)|)
              /(2 × c_j + 1)
        p_j := argmin(h(p_j) > 0), ∀p_j ∈ P
        l_k := a lecturer who offers p_j
        if |M(p_j)| < c_j and |M(l_k)| < d_k then
            M := M ∪ {(s_i, p_j)}
        else if |M(p_j)| = c_j then
            [s_t, g(s_t)] := Choose_Student(M(p_j), l_k)
            if g(s_t) > n + 1 or rank(l_k, s_i) < rank(l_k, s_t) then
                M := M \ {(s_t, p_j)} ∪ {(s_i, p_j)}
                if g(s_t) < n + 1 then
                    rank(s_t, p_j) := 0
            else
                rank(s_i, p_j) := 0
        else
            [s_w, g(s_w)] := Choose_Student(M(l_k), l_k)
            if g(s_w) > n + 1 or rank(l_k, s_i) < rank(l_k, s_w) then
                M := M \ {(s_w, p_u)} ∪ {(s_i, p_j)}, where p_u = M(s_w)
                Repair(p_u, l_k)
                if g(s_w) < n + 1 then
                    rank(s_w, p_u) := 0
            else
                rank(s_i, p_j) := 0
    return M;
end function
```

Next, for each student $s_i \in \mathcal{S}$, s_i propose to p_j if p_j is *full* or l_k is *full*, then HAG defines the function $g(s_t)$ in Algorithm 4.3 to choose student s_t corresponding to the maximum value of $g(s_t)$.

$$g(s_t) = rank(l_k, s_t) + t(s_t) + r(s_t)/(q + 1). \tag{4.3}$$

Algorithm 4.3: Heuristic function for choosing a student

Input: A set of student X.
Output: A student s_t and $g(s_t)$.

function `Choose_Student` (X, l_k)
  for *each* $s_t \in X$ **do**
    $t(s_t) := 0$;
    for *each* $p_u | rank(s_t, p_u) = rank(s_t, M(s_t))$ **do**
      $l_z :=$ a lecturer who offers p_u;
      $t(s_t) = t(s_t) + min(d_z - |M(l_z)|, 1) \times min(c_j - |M(p_u)|, 1) \times n$;
    $r(s_t) :=$ the number of projects is ranked by s_t;
    $g(s_t) := rank(l_k, s_t) + t(s_t) + r(s_t)/(q + 1)$;
  $s_t := mathrmargmax(g(s_t))$;
  return $s_t, g(s_t)$;
end function

Algorithm 4.4: Breaking blocking pairs satisfying type of *(3bi)*

Input: A matching M
Output: A matching M.

function `Repair` (p_u, l_k)
  $f :=$ true;
  while $f = true$ **do**
    $f :=$ false;
    if $s_k \in M(l_k)$ *and* $rank(s_k, p_u) < rank(s_k, p_z) | p_z = M(s_k)$ **then**
      $M := M \setminus \{(s_k, p_z)\} \cup \{(s_k, p_u)\}$;
      $p_u := p_z$;
      $f :=$ true;
  return M;
end function

The algorithm 4.4 is used to break blocking pairs when a project p_u is removed from M. For each $s_k \in M(l_k)$, if $rank(s_k, p_u) < rank(s_k, p_z)$, where $p_z = M(s_k)$, the algorithm removes (s_k, p_z) and adds (s_k, p_u) to M. This process repeats for each deleted project until it cannot form blocking pairs. If a stable matching is found that has not reached its maximum size, HAG calls Algorithm 4.5 to improve the size of the M by proposing a heuristic function. The HAG algorithm stops when it finds a perfect matching or all unassign students cannot find any projects to match.

Algorithm 4.5: Escape local minimum

```
Input: A stable matching M.
Output: A stable matching M.
function Escape(M)
    for each unassigned s_u ∈ U do
        recover s_u's original rank list;
        while s_u's rank list is non-empty do
            p_z := argmin(rank(s_u, p_z) > 0), ∀p_z ∈ P;
            l_k := a lecturer who offers p_z;
            for (each s_i ∈ M(l_k) | rank(l_k, s_i) = rank(l_k, s_u)) do
                if (|M(p_z)|) < c_z) or (s_i ∈ M(p_z) and |M(p_z)| = c_z)
                  then
                    if v(s_u) ≥ v(s_i) then
                        p_j := M(s_i);
                        M := M \ {(s_i, p_j)} ∪ {(s_u, p_z)};
                        v(s_u) := v(s_u) + 1;
                        Repair(p_j, l_k);
                        break;
            if M(s_u) ≠ ∅ then
                break;
            else
                rank(s_u, p_z) := 0;
    return M;
end function
```

We implemented these algorithms by Matlab R2019a software on a system with Xeon-R Gold 6130 CPU 2.1 GHz computer with 16 GB RAM. The experimental results show that our algorithm is much more efficient than the APX algorithm in terms of execution time and solution quality for MAX-SPA-ST of large sizes.

CONCLUSION

The thesis has completed the research objectives set out and achieved the following results:
- Proposed 02 algorithms to solve MAX-SMTI problem are published in Chapter 2.
- Proposed 02 algorithms to solve MAX-HRT problem are published in Chapter 3
- Proposing 02 algorithms to solve the problem SPA-P and SPA-ST published in Chapter 4.

Research are published in conference proceedings, and prestigious journals specializing in Heuristic, Computer Science, and Artificial Intelligence domestically and internationally

From the obtained results and limitations in this thesis, in the future, the thesis will continue to study effective heuristic algorithms for other variations of the stable marriage problem and other approaches to the problem. matrimonial stability problems and variations.