AN APPROXIMATE SEARCH ALGORITHM FOR THE STUDENT-INTERNSHIP ALLOCATION PROBLEM

Nguyen Quang Ninh [(1)], Dinh Van Nam [(2)], Hoang Huu Viet [(1)]
[1] *School of Engineering and Technology, Vinh University, Vietnam*
[2] *Faculty of Pedagogy, Ha Tinh University, Vietnam*
Received on 07/9/2022, accepted for publication on 21/10/2022
DOI: https://doi.org/10.56824/vujs.2022nt22

Abstract: This paper proposes an approximate search algorithm to solve the student-internship allocation problem. The key idea of the algorithm is that in each iteration, each student unassigned to an enterprise will be assigned to an enterprise where the student ranks it highest, and it remains at maximum capacity. If the enterprise assigned to the student overcomes its capacity, then the enterprise will remove a student to whom it ranks the worst student to allow it not to overcome its capacity. Experimental results with randomly generated datasets show that our algorithm is efficient for the problem of large sizes.

Keywords: Blocking pair; stable matching; internship allocation; approximation algorithm.

1. Introduction

In the trend of university training associated with enterprises, the assignment of students to internship enterprises is a frequent problem to be solved in universities. This problem is normally solved in one of the following ways: *(i)* training departments directly assign students to internship enterprises; or *(ii)* the student applies for a certificate of acceptance letter of the internship from a certain enterprise and the training department considers accepting internships for students. However, each enterprise is often limited in the number of students admitted to the internship. In case there are too many students applying for an internship at a certain enterprise at the same time, the above solution is not effective. The reason is that it is very difficult to satisfy both the requirement of choosing an intern enterprise and the requirement of choosing an intern of the enterprise.

The assignment of students to internship enterprises to is a one-to-many problem and is similar to the problem *"The Hospitals/Residents Problem"*, abbreviated as the HR problem [1], [2], where students refer as *"residents"* and enterprises refer as *"hospitals"*. To be consistent with the symbols in the HR problem, the symbols for *"residents"* and *"hospitals"* will be used to represent the symbols for students and enterprises. The HR problem is described as consisting of a set $R = \{r_1, r_2, \ldots, r_n\}$ of students, a set $H = \{h_1, h_2, \ldots, h_m\}$ of enterprises, in which *(i)* each student $r_i \in R$ ranks a subset of H in a strict priority order (i.e. no equal priority exists); *(ii)* each enterprise $h_j \in H$ ranks a subset of R in a strict priority order; and (*iii*) each enterprise $h_j \in H$ has an integer $c_j \in Z^+$ to indicate the maximum number of students that h_j can accept interns. The requirement of the HR problem is to find a stable match M of students and enterprises such that M has no blocking pair, in which a pair $(r_i, h_j) \in R \times H$ is said to be a block pair for matching M if: *(i)* r_i ranks h_j and vice versa; (*ii*) r_i is not matched to any enterprise or ranks h_j higher than the enterprise matched to r_i; and (*iii*) h_j is not enough students to be matched or h_j ranks r_i higher than the lowest ranked enterprise that is matched for h_j in matching M.

Email: viethh@vinhuni.edu.vn (H. H. Viet)

The HR problem requires students to rank enterprises in a strict priority order, so it is difficult to apply to real-world problems. Therefore, some extensions of the HR problem have been proposed [2]-[7], in which the HR problem does not require students/enterprises to provide a strict priority ranking list *(Hospitals/Residents problem with Ties)* referred to as HRT [3], [5], [7] received the most attention. With the permission of students and enterprises to expand lists of rankings with equal priority, the definitions of stability matching include weak stability, strong stability, and super-stability [3]. Given an instance I of the HRT problem, Irving et al. have shown that the instance I can exist many weakly stable matchings with different sizes [4]. Therefore, in order to have the maximum number of students assigned to the internship enterprises, it is necessary to ensure that the matching is not only stable, but also that the largest number of students are assigned to the enterprises. This is the problem of finding the weakly stable matching with the largest size, called the MAX-HRT problem, and has been shown by Iwama et al. to be an NP-hard problem even if: *(i)* each enterprise $h_j \in H$ has $c_j = 1$; and *(ii)* the same priority ranking appears only in the lists of students or in the lists of enterprises [9].

In this paper, we propose an approximation search algorithm to solve the MAX-HRT problem. Experimental results on randomly generated datasets show that the proposed algorithm effectively solves the large MAX-HRT problem in terms of both execution time and solution quality. Hereafter, the MAX-HRT problem will be called the HRT problem. The rest of the paper is organized as follows: Section 2 presents the HRT problem and some related works; Section 3 describes the proposed algorithm for the HRT problem; Section 4 presents the experimental results on randomly generated datasets and Section 5 gives the conclusions of the paper.

2. HRT problem and related works

2.1. HRT problem

In this section, the HRT problem will be presented [3], [5] and an example will be given to illustrate the concepts involved in the problem. An instance I of the HRT problem consists of a set $R = \{r_1, r_2, \ldots, r_n\}$ of students and a set $H = \{h_1, h_2, \ldots, h_m\}$ of enterprises, where: (*i*) each student $r_i \in R$ ranks a subset of H in a non-descending order in her/his ranking list; (*ii*) each enterprise $h_j \in H$ ranks a subset of R in a non-descending order in its ranking list; (*iii*) each enterprise $h_j \in H$ offers a maximum number of students $c_j \in Z^+$ that can accept internships. The notation $A = \{(r_i, h_j) \in R \times H\}$ is a set of acceptable pairs, where r_i ranks h_j in r_i 's prioritized list and vice versa. The notation $rank(r_i, h_j)$ is the rank order of h_j in the priority sorted list of r_i and $rank(h_j, r_i)$ is the rank order of r_i in the priority sorted list of h_j. If a student r_i prioritizes enterprise h_j higher than an enterprise h_k, then we denote $rank(r_i, h_j) < rank(r_i, h_k)$. If a student r_i prioritizes two enterprises h_j and h_k equally, we denote $rank(r_i, h_j) = rank(r_i, h_k)$. The same rating symbols are also used for enterprises.

Definition 1 (*Matching*): A matching M is a set of pairs $(r_i,h_j) \in A$, where each student $r_i \in R$ can match at most one enterprise $h_j \in H$ and each enterprise h_j can match at most c_j students.

If a pair $(r_i, h_j) \in M$, we denote $M(r_i) = h_j$, $M(h_j) = \{r_i \mid (r_i, h_j) \in M\}$ and $|M(h_j)|$ is the number of students assigned to the enterprise h_j. If a student r_i is not matched for any

enterprises, we denote $M(r_i) = \emptyset$. An enterprise $h_j \in H$ is said to be *under-subscribed*, *full*, or *over-subscribed* for the number of students if $|M(h_j)| < c_j$, $|M(h_j)| = c_j$, or $|M(h_j)| > c_j$, respectively.

Definition 2 (*Blocking pair*): A pair $(r_i, h_j) \in R \times H$ is said to be a blocking pair for a matching M if the following conditions are satisfied: (*i*) $(r_i, h_j) \in A$; (*ii*) $M(r_i) = \emptyset$ or $rank(r_i, h_j) < rank(r_i, M(r_i))$; (*iii*) $|M(h_j)| < c_j$ or $rank(h_j, r_i) < rank(h_j, r_w)$ in which r_w is the student for whom h_j ranks lowest in $M(h_j)$.

Definition 3 (*Stable matching*): A matching M of an instance I is said to be a stable matching if there are no blocking pairs for M, otherwise M is called unstable matching.

We denote $|M|$ by the number of students assigned in the stable matching M. A stable matching M is said to be a perfect matching if $|M| = n$, that is, every student $r_i \in R$ is matched to the enterprises, otherwise M is said to be non-perfect.

An illustrative example of an instance I of the HRT problem with 8 students and 4 enterprises is described in Table 1, in which the order of equal priority is placed in pairs of "()". For example, the notation r_5: (h_1, h_2) h_3 means that r_5 ranks h_1 and h_2 equally, but ranks h_1 and h_2 higher than h_3, i.e. $rank(r_5, h_1) = rank(r_5, h_2) = 1$ and $rank(r_5, h_3) = 2$. The matching $M = \{(r_1, h_1), (r_2, h_1), (r_3, h_2), (r_4, h_3), (r_6, h_4), (r_7, h_2)\}$ is an unstable matching because there exist several blocking pairs for M including (r_3, h_1), (r_5, h_1), (r_5, h_2), (r_5, h_3), and (r_8, h_2). The matching $M = \{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_3), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_2)\}$ is a stable matching since there are no blocking pairs for M and moreover, M is a perfect matching because $|M| = 8$.

Table 1: *An example of an instance I of the HRT problem*

List in order of priority of students	*List in order of priority of enterprises*
r_1: h_1 h_3	h_1: r_3 (r_2 r_5) (r_1 r_7 r_8)
r_2: (h_1 h_4)	h_2: r_3 r_8 (r_5 r_6) r_7 r_4
r_3: h_1 h_2	h_3: (r_1 r_4 r_6) r_5
r_4: (h_2 h_3 h_4)	h_4: r_2 r_6 r_4
r_5: (h_1 h_2) h_3	
r_6: h_4 h_2 h_3	*Capacities of enterprises*:
r_7: h_1 h_2	$c_1 = 2$, $c_2 = 2$, $c_3 = 2$, và $c_4 = 2$.
r_8: (h_1 h_2)	

2.2. Some related works

In recent years, most of the proposed algorithms to solve the HRT problem are approximation algorithms because HRT is an NP-hard problem. An algorithm is said to be r-approximate if it always finds a stable match M with $|M| \geq |M_{opt}|/r$ for all instances of the HRT problem, where M_{opt} is a stable matching of the largest size. Manlove et al. [9] proposed a 2-approximation algorithm to find a weakly stable matching with the largest size for the HRT problem. Irving and Manlove [5] have proposed two heuristic search algorithms for the HRT problem, in which the equal priority ranking appears only in the priority list of enterprises. Király [11] proposed two 3/2-approximation algorithms with linear time, in which an algorithm is applied to the HRT problem when equal priority appears only in the ranked list of enterprises and an algorithm is applied to the general

HRT problem. Kwanashie et al. have proposed an integer programming model, abbreviated as IP, to solve the HRT problem [12]. The basic idea of the IP algorithm for the HRT problem includes: (*i*) remove the acceptable pairs in the priority lists of students and enterprises that do not belong to the stable matchings; and (*ii*) use the CPLEX IP tool to solve the HRT problem. Munera et al. have proposed an adaptive search algorithm, abbreviated as AS, to solve the HRT problem [7]. The basic idea of the AS algorithm is to convert the HRT problem to the SMTI problem and apply the adaptive search algorithm to the SMTI problem to solve the HRT problem [13].

Algorithm 1: HS algorithm for the HRT problem

1. **function** HS(I)
2. $M := \emptyset$;
3. $a(r_i) := 1, \forall r_i \in R$;
4. $h(h_j, r_i) := 0, \forall (r_i, h_j) \in A$;
5. **while** ($\exists r_i \in R \mid a(r_i) = 1$) **do**
6. **if** ($rank(r_i, h_j) = 0, \forall h_j \in H$) **then**
7. $a(r_i) := 0$;
8. continue;
9. **end**
10. $H_j := \text{argmin}(rank(r_i, h_j)), \forall h_j \in H$ and $rank(r_i, h_j) > 0$;
11. $h_j := \text{argmax}(c_j - |M(h_j)|), \forall h_j \in H_j$;
12. $M := M \cup \{(r_i, h_j)\}$;
13. $a(r_i) := 0$;
14. $f(j) :=$ frequency of $j = 1, 2, \ldots, m$ in $rank(r_i, h_j), \forall h_j \in H$;
15. $h(h_j, r_i) := rank(h_j, r_i) + (m \times max(f(j)) + sum(f(j))) / (m^2 + 1), \forall j = 1, ..., m$;
16. **if** ($|M(h_j)| > c_j$) **then**
17. $r_w := argmax(h(h_j, r_i)), \forall r_i \in M(h_j)$;
18. $M := M \setminus \{(r_w, h_j)\}$;
19. $rank(r_w, h_j) := 0$;
20. $a(r_w) := 1$;
21. $h(h_j, r_w) := 0$;
22. **end**
23. **end**
24. **return** M;

3. Proposed algorithm

In this section, an approximation algorithm in the form of heuristic search, abbreviated as HS, is proposed to solve the problem of assigning students to internship enterprises.

3.1. Algorithm

The HS algorithm to solve the HRT problem is proposed as in Algorithm 1. During the execution of the algorithm, each student $r_i \in R$ is set to one of two states, either paired with a certain enterprise in a stable matching M (i.e., $a(r_i) = 1$) or unpaired with any enterprise (i.e., $a(r_i) = 0$).

First, the algorithm establishes all students in the unpaired state in the matching M. In addition, the value of the heuristic function is also set $h(h_j, r_i) = 0$ for every pair $(r_i, h_j) \in A$. At each iteration, the algorithm runs the following: The algorithm first finds a student $r_i \in R$ that has not been matched with any enterprises. If r_i has selected all enterprises h_j in r_i's ranking list, then r_i changes to unpaired state, i.e., student r_i will not be matched with any enterprises and the algorithm continues for other students (lines 6-9). On the contrary, the algorithm finds a set of enterprises H_j that r_i ranks the highest priority and then chooses an enterprise $h_j \in H_j$ that has the most internships (lines 10-11). Next, the algorithm assigns student r_i to enterprise h_j in the matching M and sets the state r_i to be matched. When student r_i is paired with enterprise h_j, the algorithm updates the value of the heuristic function $h(h_j, r_i)$ for the pair (h_j, r_i) to eliminate a "worst" student in the set $M(h_j)$ of students who have been matched with enterprise h_j (line 14-15). If the enterprise h_j exceeds the maximum number of students c_j to accept the internship, the algorithm finds and removes a worst student r_w in the set $M(h_j)$ corresponding to the maximum value $h(h_j, r_w)$, h_j will be removed from r_w's priority ranking list. At the same time, the algorithm will set the state of student r_w as unpaired and the assigned heuristic function value $h(h_j, r_w) = 0$ (lines 16-22).

The algorithm iterates until every student has a matched state, i.e., has been matched at least once with a certain enterprise $h_j \in H$ and returns a stable matching M. Note that a student r_w corresponding to the maximum of the function $h(h_j, r_w)$ (line 15) means that $rank(h_j, r_i)$ is the largest and $m \times max(f(j)) + sum(f(j)) / (m^2 + 1)$ is the largest. Furthermore, we have $rank(h_j, r_i) \geq 1$ and $0 < (m \times max(f(j)) + sum(f(j))) / (m^2 + 1) < 1$, so that eliminating a student r_w corresponding to the maximum of the function $h(h_j, r_w)$ will ensure that the algorithm does not form blocking pairs for M and student r_w will have at most the chances of being matched with the most remaining enterprises because r_w not only ranks in priority with the most enterprises (i.e. $m \times max(f(j))$) but also ranks the most enterprises (i.e. $sum(f(j))$).

3.2. Example

Consider an example of an instance I of the HRT problem given in Table 1 with the ranking lists of students and enterprises described in Table 2. The iteration steps of the algorithm to find a perfect matching are shown in Table 3.

From iteration steps 1 to 6, students $r_1, r_2, \ldots, r_6$ selects and is paired with the highest priority enterprises, i.e., $M = \{(r_1, h_1), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4)\}$. At iteration step 7, r_7 chooses h_1, we have $h(h_1, r_7) = 3.4$ and $M = \{(r_1, h_1), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1)\}$. However, because h_1 exceeds the number of students $c_1 = 2$, the algorithm will eliminate one of the three pairs (r_1, h_1), (r_3, h_1) and (r_7, h_1). Since

$h(h_1, r_1)$ has the largest value, the pair (r_1, h_1) is excluded from M, that is, $M = \{(r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1)\}$ and $h(r_1, r_1) = 0$. At iteration step 8, r_1 chooses h_3, we have $h(h_3, r_1) = 1.6$ and $M = \{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1)\}$. At iteration step 9, r_8 chooses h_1, we have $h(h_1, r_8) = 4.2$ and $M = \{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_1)\}$. However, because h_1 exceeds the number of students $c_1 = 2$, the algorithm will eliminate one of the three pairs (r_3, h_1), (r_7, h_1), (r_8, h_1). Since $h(h_1, r_8)$ has the largest value, the pair (r_8, h_1) will be removed from M and we have the matching $M = \{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1)\}$. At iteration step 10, r_8 chooses h_2, we have $h(h_2, r_8) = 2.6$ and $M = \{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_2)\}$. Since $h(h_2, r_4)$ has the largest value of 3 pairs (r_4, h_2), (r_5, h_2), (r_8, h_2), it will be eliminated to obtain the matching $M = \{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_2)\}$. At iteration step 11, r_4 chooses h_3 and just enough students are matched $c_3 = 3$, the algorithm returns a perfect matching $M = \{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_3), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_2)\}$.

Table 2: *Rank list of instance I of students and enterprises*

List in order of priority of students

	h_1	h_2	h_3	h_4
r_1:	1	0	2	0
r_2:	1	0	0	1
r_3:	1	2	0	0
r_4:	0	1	1	1
r_5:	1	1	2	0
r_6:	0	2	3	1
r_7:	1	2	0	0
r_8:	1	1	0	0

List in order of priority of enterprises

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
h_1:	3	2	1	0	2	0	3	3
h_2:	0	0	1	5	3	3	4	2
h_3:	1	0	0	1	2	1	0	0
h_4:	0	1	0	3	0	2	0	0

Maximum number of students of enterprises: $c_1 = 2$, $c_2 = 2$, $c_3 = 2$, và $c_4 = 2$.

Table 3: *The iteration steps of the HS algorithm for the example in Table 1*

Step	r_i	h_j	$h(h_j, r_i)$	M
1	r_1	h_1	3.4	$\{(r_1, h_1)\}$
2	r_2	h_4	2.2	$\{(r_1, h_1), (r_2, h_4)\}$
3	r_3	h_1	1.4	$\{(r_1, h_1), (r_2, h_4), (r_3, h_1)\}$
4	r_4	h_2	6.8	$\{(r_1, h_1), (r_2, h_4), (r_3, h_1), (r_4, h_2)\}$
5	r_5	h_2	4.0	$\{(r_1, h_1), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2)\}$
6	r_6	h_4	2.2	$\{(r_1, h_1), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4)\}$
7	r_7	h_1	3.4	$\{(r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1)\}$
8	r_1	h_3	1.6	$\{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1)\}$
9	r_8	h_1	4.2	$\{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_2), (r_5, h_2), (r_6, h_4), (r_7, h_1)\}$
10	r_8	h_2	2.6	$\{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_2)\}$
11	r_4	h_3	2.2	$\{(r_1, h_3), (r_2, h_4), (r_3, h_1), (r_4, h_3), (r_5, h_2), (r_6, h_4), (r_7, h_1), (r_8, h_2)\}$

4. Some experimental results

In this section, we describe the experiments to evaluate the efficiency of the proposed algorithm. The quantity and execution time to find perfect matchings are compared between the proposed algorithm and the AS algorithm [7]. All experiments were performed by Matlab software version 2019a running on a Core i7-8550U personal computer with 1.8GHz processor and 16Gb memory.

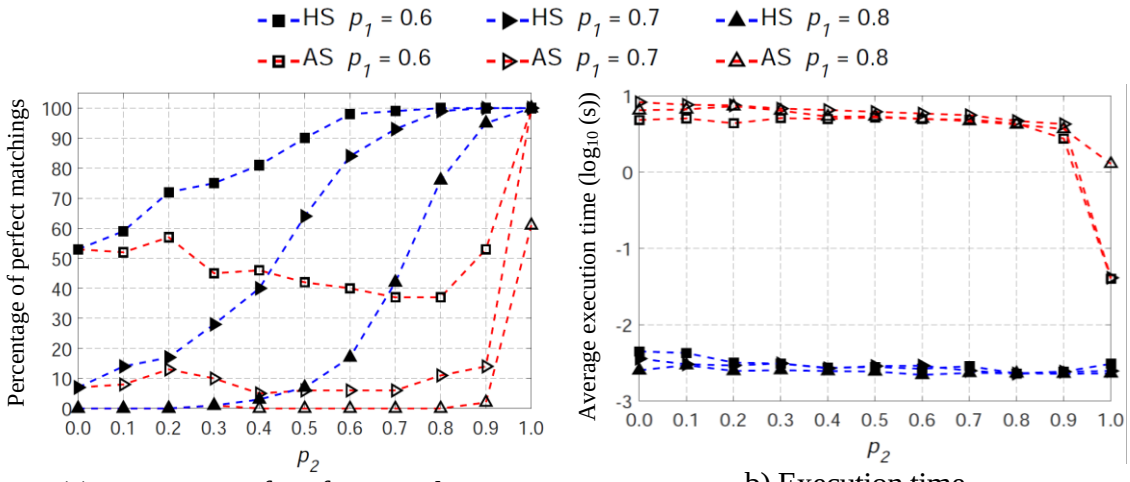
We extend the procedure for creating the problem SMTI [14] (a special case of the HRT problem with $n = m$ and $c_j = 1$, $\forall h_j \in H$) to generate instances of HRT with the parameter set (n, m, $\{c_1, c_2, \ldots, c_m\}$, p_1, p_2), where n is the number of students, m is the number of enterprises, c_j ($j = 1, 2, \ldots, m$) is the maximum number of students that an enterprise $h_j \in H$ can accept an internship, p_1 is the probability of the appearance of enterprises and students in the ranked list of $r_i \in R$ and $h_j \in H$, p_2 is the probability that $r_i \in R$ and $h_j \in H$ rank equally among enterprises and students. This means that each student ranks about $m \times (1 - p_1)$ enterprises and each enterprise ranks about $n \times (1 - p_1)$ students in each instance created.

In addition, since the stable matchings contain only acceptable pairs $(r_i, h_j) \in A$, we therefore create instances where the ranking lists of students and enterprises consist of only acceptable pairs. Table 2 illustrates a randomly generated instance I according to the priority ranking list of the HRT problem with parameters (8, 4, {2, 2, 2, 2}, 0.5, 0.5).

Experiment 1: In this experiment, we choose $n = 200$, $m = 20$, $p_1 \in \{0.1, 0.2, \ldots, 0.8\}$ and $p_2 \in \{0.0, 0.1, \ldots, 1.0\}$. This means that when p_1 increases from 0.1 to 0.8, each student ranks from 18 enterprises down to 4 enterprises, and each enterprise ranks from 180 students down to 40 students. For each combination of the values of the parameters (n, m, p_1, p_2), we randomly generate 100 instances where $c_j = n/m$ for all $h_j \in H$ ($j = 1, 2, \ldots, m$). With this setting we have $\sum_{j=1}^{m} c_j = n$, i.e., each student will have the possibility of getting an internship from an enterprise. Obviously if $\sum_{j=1}^{m} c_j < n$, at least one student will not get the internship, that is, we cannot find a perfect matching. In addition, we set the maximum number of iterations in the AS algorithm to be 5000.

Figure 1(a) shows the percentage of perfect matchings found by the HS and AS algorithms. Note that when p_1 increases from 0.1 to 0.5, both the HS and AS algorithms find 100% of perfect matchings, so we do not show the experimental results on this figure. The experimental results show that when p_1 increases from 0.6 to 0.8, the percentage of perfect matchings found by HS and AS both decrease because as the number of enterprises prioritized by students and the number of students ranked by enterprises decreases, which leads to a decrease in the number of acceptable pairs and thus it is difficult for HS and AS algorithms to find perfect matchings.

When $p_2 = 0$, that is, the priority ranking list of enterprises and students has no equal priority, the HS and AS algorithms both find the same number of perfect matchings because all stable matchings have the same size. As p_2 increases from 0.1 to 1.0, HS finds more perfect matchings than AS.



(a) Percentage of perfect matchings

b) Execution time for finding perfect matchings

Figure 1: *The result of Experiment 1*

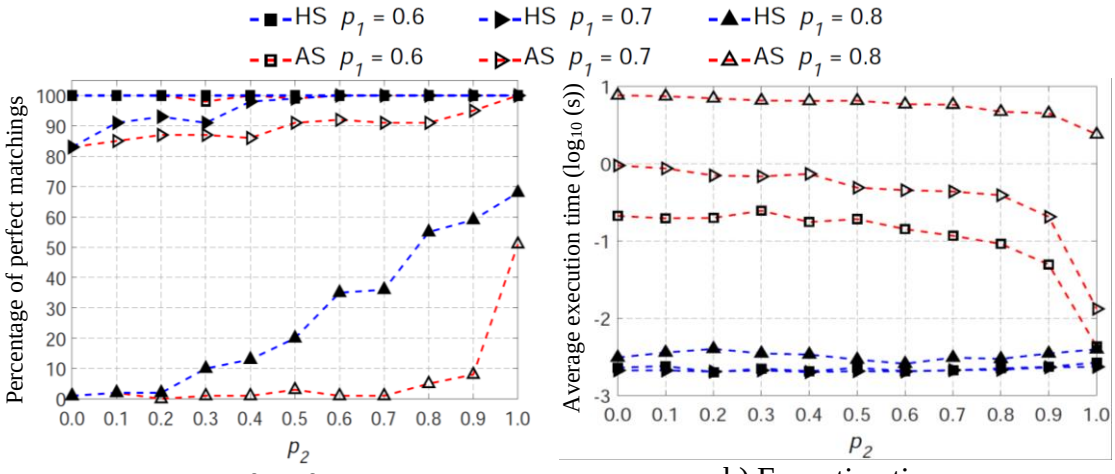
Figure 1(b) shows the average execution time to find the perfect matchings of the HS and AS algorithms. When p_1 increases from 0.6 to 0.8, the execution time to find the perfect matchings of HS and AS does not change significantly. When p_2 increases from 0.0 to 0.9, the execution time to find perfect matchings of HS and AS is almost unchanged. When $p_2 = 1.0$, the execution time of HS to find perfect matchings is approximately the same as that of perfect matchings when $p_2 = 0.9$, while the execution time of AS to find perfect matchings is slightly reduced. In addition, the execution time of HS to find perfect matchings increased from $10^{-2.5}$ seconds to $10^{-2.1}$ seconds, while the execution time of AS to find perfect matchings increased from $10^{-1.4}$ seconds to $10^{0.9}$ seconds, that is, HS finds perfect matchings 12 to 1000 times faster than AS.

Experiment 2: In this experiment, we choose the values of the parameters n, m, p_1 and p_2 as in Experiment 1. For each combination of the values of the parameters (n, m, p_1, p_2), 100 instances of the HRT problem were randomly generated, where the maximum number of students c_j for each enterprise $h_j \in H$ ($j = 1, 2, \ldots, m$) is chosen as a random integer in the interval $[0.1q, 0.4q]$, in which q is the total number of students ranked by the enterprise $h_j \in H$. This can be understood that each enterprise $h_j \in H$ ranks q students but only selects from 10% to 40% of interns.

Figure 2 shows the percentage of perfect matchings and the average execution time to find the perfect matchings of the HS and AS algorithms. Figure 2(a) shows that when $p_1 = 0.6$ or $p_1 = 0.7$, both HS and AS find a higher number of perfect matchings than in the case of $c_j = n/m$ as in Experiment 1. In addition, the HS algorithm finds a much higher percentage of perfect matchings than the AS algorithm. Figure 2(b) shows that the HS algorithm has a much smaller average time to find perfect matchings than the AS algorithm, i.e. HS runs much faster than the AS algorithm.

Experiment 3: In this experiment, the number of students and the number of enterprises is changed to consider the performance of HS and AS algorithms. We choose $n = 300$, $m \in \{15, 20, 25\}$, $p_1 = 0.7$ and $p_2 \in \{0.0, 0.1, \ldots, 1.0\}$, that is, each student ranks about 5 to 8 enterprises and each enterprise ranks about 90 students. For each combination

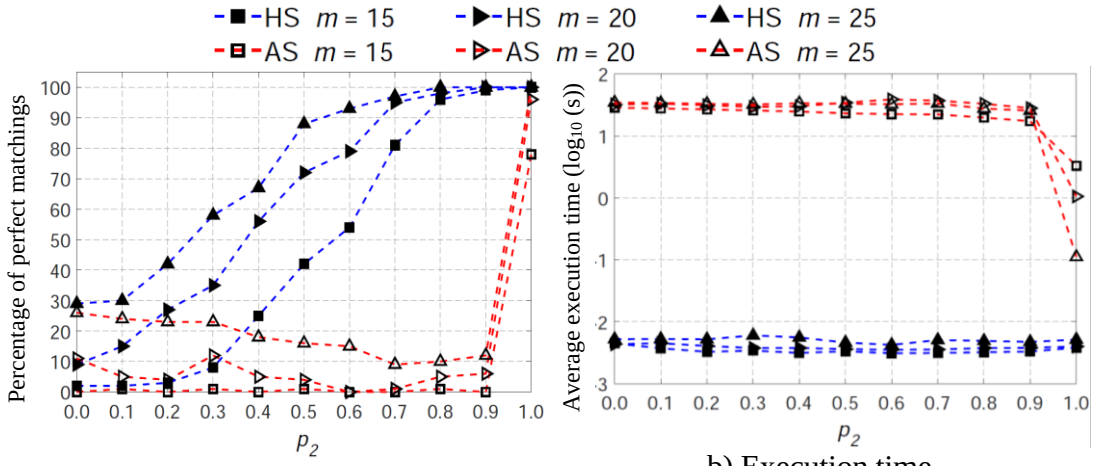
of values of the parameters (n, m, p_1, p_2), 100 instances of the HRT problem were randomly generated, in which the maximum number of students of enterprises $h_j \in H$ (j = 1, 2, . . ., m) is $c_j = n/m$.



a) Percentage of perfect matchings

b) Execution time for finding perfect matchings

Figure 2: *The result of Experiment 2*



a) Percentage of perfect matchings

b) Execution time for finding perfect matchings

Figure 3: *The result of Experiment 3*

Figure 3(a) shows the percentage of perfect matchings found by the HS and AS algorithms. As m increases from 15 to 25, it is easier for both HS and AS to find perfect matchings because the generated instances have more acceptable pairs in the priority ranking lists of both students and enterprises. In addition, the results show that HS finds a much higher number of perfect matchings than AS.

Figure 3(b) shows the average execution time to find the perfect matchings of the HS and AS algorithms. The average time to find the perfect matchings of AS ranges from 10^{-1} = 0.1 seconds (when p_2 = 1.0) to $10^{1.5}$ = 31.6 seconds (when p_2 = 0.0, 0.1, . . ., 0.9), while the average time to find the perfect matchings of HS is about $10^{-2.3}$ = 0.005 seconds,

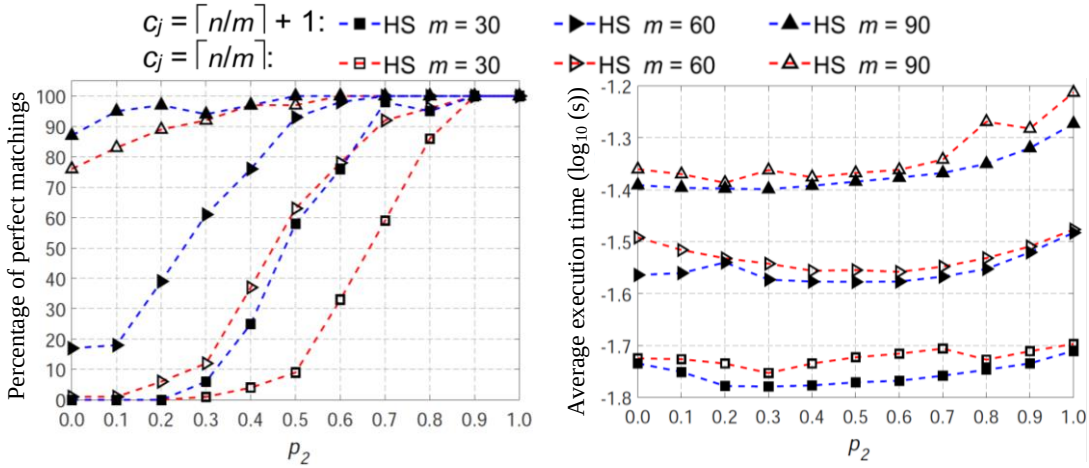
that is, the student finds the perfect matching faster than AS about 20 (when $p_2 = 0.1$) to 6300 times (when $p_2 = 0.0, 0.1, \ldots, 0.9$).

With the above three experiments, we see that the HS algorithm not only outperforms the AS algorithm in the number of perfect matchings found, but also performs many times faster than AS when finding the perfect matchings.

Experiment 4: In this experiment, we consider the efficiency of HS algorithm when n and m are large: $n = 1000$, $m \in \{30, 60, 90\}$, $p_1 = 0{,}9$, and $p_2 \in \{0.0, 0.1, \ldots, 1.0\}$. For each combination of the values of the parameters (n, m, p_1, p_2), we randomly generate 100 instances with $c_j = \lceil n/m \rceil$ and 100 instances with $c_j = \lceil n/m \rceil + 1$.

Figure 4(a) shows that as the number of enterprises increases, it is easier for students to find perfect matchings because the number of acceptable pairs in the generated instances increases. In addition, students found higher perfect matchings as each enterprise increased its likelihood of accepting an additional intern. Figure 4(b) also shows that the average time for students to find perfect matchings is about $10^{-1.7} = 0.02$, $10^{-1.5} = 0.3$, and $10^{-1.3} = 0.05$ seconds when $m = 30$, 60, and 90, respectively.

With the above experimental results, we see that the HS algorithm is not only efficient in finding the perfect matchings, but also in terms of execution time for the large HRT problem.



(a) Percentage of perfect matchings

b) Execution time for finding perfect matchings

Figure 4: *The result of Experiment 4*

5. Conclusion

In this paper, we propose an approximate heuristic search algorithm to solve the problem of assigning internship locations to students. In each iteration of the algorithm, each student who has not been matched with an enterprise will be matched with an enterprise that the student has the highest priority rating and has the most internships left. If the enterprise that is matched with students exceeds the number of students who are likely to receive an internship, the enterprise will remove a matched student whose enterprise ranks the lowest, and the student who ranks with the most enterprise-ranked

listings. This not only ensures that the enterprise does not exceed the number of interns, but also creates the most opportunities for the eliminated students to match with other enterprises. Experimental results on randomly generated datasets show that our algorithm is not only more efficient than AS algorithm in terms of execution time and solution quality, but also effective for large size problems.

REFERENCES

[1] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 9, no. 1, pp. 9-15, 1962.

[2] D. Gusfield and R.W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, 1989.

[3] R. W. Irving, D. F. Manlove, and S. Scott, "The hospitals/residents problem with ties," in *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory*, Bergen, Norway, 6/2000, pp. 259-271.

[4] R. W. Irving, D. F. Manlove, and S. Scott, "The stable marriage problem with master preference lists," *Discrete Applied Mathematics*, vol. 156, no. 15, pp. 2959-2977, 2008.

[5] R. W. Irving and D. F. Manlove, "Finding Large Stable Matchings," *Journal of Experimental Algorithmics*, vol. 14, no. 1, pp. 1-27, 2009.

[6] P. Biró, D. F. Manlove, and I. McBride, "The hospitals/residents problem with couples: Complexity and integer programming models," in *Proceeding of SEA 2014: 13th International Symposium on Experimental Algorithms*, Copenhagen, Denmark, 7/2014, pp. 10-21.

[7] D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, and P. Codognet, "A local search algorithm for SMTI and its extension to HRT problems," in *Proceedings of the 3rd International Workshop on Matching Under Preference*s, University of Glasgow, UK, 4/2015, pp. 66-77.

[8] D. F. Manlove, I. McBride, and J. Trimble, "almost-stable" matchings in the hospitals/ resident's problem with couples," *Constraints Journal*, vol. 22, no. 1, pp. 50-72, 2017.

[9] K. Iwama, D. Manlove, S. Miyazaki, Y. Morita, "Stable marriage with incomplete lists and ties," in P*roceedings of ICALP 1999: The 26th International Colloquium on Automata, Languages, and Programming*, 1999, pp. 443-452.

[10] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, "Hardvariants of stable marriage," *Theoretical Computer Science*, vol. 276, no. 1, pp. 261-279, 2022.

[11] Z. Király, "Linear time local approximation algorithm for maximum stable marriage," *Algorithms*, vol. 6, no. 1, pp. 471-484, 2013.

[12] A. Kwanashie, D. F. Manlove, "An integer programming approach to the hospitals/residents problem with ties," in *Proceedings of the International Conference on Operations Research*, Erasmus University Rotterdam, 2013, pp. 263-269.

[13] D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, P. Codognet, "Solving hard stable matching problems via local search and cooperative parallelization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas, 2015, pp. 1212-1218.

[14] I. P. Gent and P. Prosser, "An empirical study of the stable marriage problem with ties and incomplete lists," in *Proceedings of the 15th European Conference on Artificial Intelligence*, Lyon, France, 2002, pp. 141-145.

TÓM TẮT

MỘT THUẬT TOÁN TÌM KIẾM XẤP XỈ CHO BÀI TOÁN PHÂN CÔNG ĐỊA ĐIỂM THỰC TẬP CHO SINH VIÊN

Nguyễn Quang Ninh [(1)], Đinh Văn Nam [(2)], Hoàng Hữu Việt [(1)]

[1] *Viện Kỹ thuật và Công nghệ, Trường Đại học Vinh, Việt Nam*

[2] *Khoa Sư phạm, Trường Đại học Hà Tĩnh, Việt Nam*

Ngày nhận bài 07/9/2022, ngày nhận đăng 21/10/2022

Bài báo này đề xuất một thuật toán tìm kiếm xấp xỉ để giải quyết bài toán phân công địa điểm thực tập cho sinh viên. Ý tưởng chính của thuật toán là trong mỗi bước lặp của thuật toán, mỗi sinh viên chưa được ghép với doanh nghiệp thực tập sẽ được ghép với một doanh nghiệp mà sinh viên xếp hạng ưu tiên cao nhất và còn nhiều chỗ thực tập nhất. Nếu doanh nghiệp được ghép với sinh viên vượt quá số lượng sinh viên có khả năng nhận thực tập, doanh nghiệp sẽ loại bỏ một sinh viên đã được ghép mà doanh nghiệp xếp hạng thấp nhất để đảm bảo không vượt quá số sinh viên thực tập. Kết quả thực nghiệm trên các bộ dữ liệu được tạo ngẫu nhiên chỉ ra rằng thuật toán của chúng tôi hiệu quả cho bài toán kích thước lớn.

Từ khóa: Cặp khối; phép ghép ổn định; phân công thực tập; thuật toán xấp xỉ.