AN EFFICIENT DEPTH-FIRST SEARCH ALGORITHM FOR SOLVING THE MAXIMUM STABLE MARRIAGE PROBLEM WITH TIES AND INCOMPLETE LISTS

Le Quoc Anh[1], Dinh Van Nam[2], Hoang Huu Viet[1,*]
[1]*Institute of Engineering and Technology, Vinh University, Nghe An, Vietnam*
[2]*Faculty of Pedagogy, Ha Tinh University, Ha Tinh, Vietnam*

ARTICLE INFORMATION

Journal: *Vinh University Journal of Science Natural Science, Engineering and Technology*
p-ISSN*: 3030-4563*
e-ISSN: *3030-4180*

Volume: 53
Issue: 4A
Correspondence:
viethh@vinhuni.edu.vn

Received: *05 October 2024*
Accepted: *06 December 2024*
Published: *20 December 2024*

Citation:
Le Quoc Anh, Dinh Van Nam, Hoang Huu Viet (2024). An efficient depth-first search algorithm for solving the maximum stable marriage problem with ties and incomplete lists. ***Vinh Uni. J. Sci.*** *Vol. 53 (4A), pp. 112-123 doi: 10.56824/vujs.2024a146a*

ABSTRACT

This paper proposes an efficient depth-first search algorithm to solve the maximum stable marriage problem with ties and incomplete preference lists. The key idea of the algorithm is to initialize an empty matching and mark all men as unmatched. In each iteration, an unmatched man proposes to the most preferred woman on his rank list. If the woman is unmatched or prefers the proposing man over her current partner, she is assigned to the man, forming a new pair in the matching. Otherwise, she keeps her current partner and rejects the proposing man. When a man is rejected by a woman, he becomes unmatched. The algorithm then recursively processes the next unmatched man until it either finds a complete matching or reaches a predefined maximum number of iterations. Experimental results on randomly generated datasets show that our algorithm is effective in producing high-quality solutions to the problem.

Keywords: Depth-first search; Gale-Shapley algorithm; stable matching; stable marriage problem; ties and incomplete lists.

1. Introduction

In 1962, Gale and Shapley introduced the Stable Marriage Problem (SMP) [1], a fundamental two-sided matching problem. An instance of SMP involves n men and n women, where each individual ranks all members of the opposite set in a strict order of preference. The objective is to find a one-to-one stable matching, ensuring that no man and woman would prefer each other over their assigned partners. Gale and Shapley also proposed the Gale-Shapley algorithm [1], which efficiently produces a stable matching for any given instance of SMP. In practical applications of the two-sided matching problem, additional complexities such as ties in preference rankings and incomplete preference lists often arise. To handle these situations, several extensions of SMP have been developed, including the Stable Marriage Problem with Incomplete Lists (SMI) [2], the Stable Marriage Problem with Ties (SMT) [3], and the Stable Marriage Problem with Ties and Incomplete Lists (SMTI) [2], [4].

This paper focuses on the SMTI problem. An SMTI instance of size n consists of two sets: $M = \{m_1, m_2, \cdots, m_n\}$ of n men and $W = \{w_1, w_2, \cdots, w_n\}$ of n women. Each individual provides a preference list that may include ties and may be incomplete. We denote $mr(m_i, w_j)$ as the rank of w_j in m_i's list and $wr(w_j, m_i)$ as the rank of m_i in w_j's list. If m_i does not rank w_j, we set $mr(m_i, w_j) = 0$. When m_i prefers w_j over w_k, we write $mr(m_i, w_j) < mr(m_i, w_k)$. If m_i ranks w_i and w_k equally, indicating a tie, we use $mr(m_i, w_j) = mr(m_i, w_k)$. Similar notations are applied to women's rankings. A pair $(m_i, w_j) \in M \times W$ is considered acceptable if both m_i and w_j rank each other on their respective preference lists. For any acceptable pair (m_i, w_j), the rankings satisfy $1 \leq mr(m_i, w_j) \leq n$ and $1 \leq wr(w_j, m_i) \leq n$ for all $m_i \in M$ and $w_j \in W$.

A matching μ in an SMTI instance is a set of disjoint acceptable pairs $(m_i, w_j) \in M \times W$. If $(m_i, w_j) \in \mu$, then m_i and w_j are considered partners in μ, denoted as $\mu(m_i) = w_j$ and $\mu(w_j) = m_i$. If $m_i \in M$ is not assigned to any woman in μ, then m_i is called as *single*, represented as $\mu(m_i) = \emptyset$. Similarly, if w_j is not assigned to any man, then w_j is called as *single*, denoted as $\mu(w_j) = \emptyset$. With ties present in the preference lists of men and women, three stability criteria for μ are defined: *weak stability*, *strong stability*, and *super-stability* [5]. The problem of finding maximum weakly stable matchings in SMTI instances has been a significant area of research for many years [2], [6], [7], [8], [9] and is also addressed in this study.

A matching μ is called *weakly stable* if it does not admit any *blocking pair*. A pair (m_i, w_j) is considered as a blocking pair for μ if all the following conditions hold: (a) $mr(m_i, w_j) > 0$ and $wr(w_j, m_i) > 0$, meaning (m_i, w_j) is an acceptable pair; (b) $\mu(m_i) = \emptyset$ or $mr(m_i, w_j) < mr(m_i, \mu(m_i))$, meaning m_i is either single or prefers w_j over his current partner; and (c) $\mu(w_j) = \emptyset$ or $wr(w_j, m_i) < wr(w_j, \mu(w_j))$, meaning w_j is either single or prefers m_i over her current partner. If any blocking pair exists, μ is considered *unstable*. The size of a weakly stable matching μ, denoted $|\mu|$, is the number of men (or women) involved in the matching. A weakly stable matching is *complete* if $|\mu| = n$, meaning all men and women are matched; otherwise, it is *incomplete*.

Since weakly stable matchings always exist in any SMTI instance and can vary in size [10], a natural goal is to find a matching of maximum size, meaning that the highest number of men and women are matched. This problem, known as Max-SMTI [2], [11], has been proven to be NP-hard [2], [4], making the development of efficient algorithms for solving large instances a significant challenge for researchers.

In this paper, we present an efficient depth-first search algorithm to address the Max-SMTI problem. Experimental results on randomly generated datasets demonstrate that the proposed algorithm achieves high solution quality for large instances of the problem. As an exhaustive algorithm, the proposed algorithm serves as a reliable benchmark for evaluating the solution quality of other methods to the maximum stable marriage problem with ties and incomplete lists. For simplicity, we will refer to the Max-SMTI problem as the SMTI problem in the remainder of this paper. The structure of the paper is as follows: Section 2 reviews related work, Section 3 details the proposed algorithm for solving the SMTI problem, Section 4 presents experimental results on randomly generated datasets, and Section 5 concludes the paper.

2. Related works

In recent years, most algorithms proposed in the literature for solving the SMTI problem have been approximate. An algorithm is called as an r - approximation for any instance of SMTI if it results in a stable matching μ such that $|\mu| \geq |\mu_{opt}|/r$, where μ_{opt} represents a maximum size stable matching [8].

Various algorithms have been developed to tackle the SMTI problem, with many extending the Gale-Shapley (GS) algorithm initially developed for the Stable Marriage Problem (SMP) [1]. GS-based approximation algorithms generally begin with an empty matching, denoted as μ, and iteratively build a maximum stable matching. In this process, each man $m_i \in M$ proposes to his most-preferred woman $w_j \in W$. If w_j is either single or prefers m_i over her current partner $m_k \in M$, she accepts m_i, forming a stable pair $(m_i, w_j) \in \mu$. Otherwise, she rejects m_i, who then removes w_j from his preference list. If m_k is rejected by w_j, he re-enters the process and removes w_j from his list. A man m_i who exhausts all women on his preference list remains permanently single.

McDermid [12] proposed a 3/2 - approximation algorithm for SMTI instances with a time complexity of $O(n^{3/2}m)$, where n is the total number of men and women and m is the total length of their preference lists. Király [13] introduced two linear - time approximation algorithms based on the GS algorithm [1]. The first, namely GSA1, is a 3/2 - approximation algorithm for the specific case of SMTI instances, where ties are present on one side only. The second, namely GSA2, is a 5/3 - approximation algorithm for the general SMTI case. Later, Király [8] reformulated GSA1 for the one-sided ties case and proposed a new 3/2 - approximation algorithm, called New Algorithm, by slightly modifying GSA1 to handle the general SMTI case. Iwama et al. [14] extended GSA1 for one-sided ties to develop a 25/17 - approximation algorithm. However, the assumption that ties occur on one side only is impractical in many scenarios. Paluch [15] modified the GS algorithm to create a 3/2 - approximation algorithm, namely GSM, with a linear runtime of $O(m)$ and additionally is simpler than that of McDermid [12], where m is the sum of the lengths of the men's and women's lists.

Local search approach has also been explored for solving the SMTI problem. Unlike GS-based approximation algorithms that construct a stable matching iteratively from an empty set, local search algorithms start with a randomly generated matching μ and iteratively refine it to improve stability. At each iteration, a set of neighboring matchings is generated, and the best matching is selected based on an objective function. Gelain et al. [7], [16] proposed a local search algorithm called LTIU to handle SMTI but did not provide an approximation ratio for their method. Munera et al. [9] applied an adaptive search algorithm, referred to as AS, to solve SMTI. More recently, we introduced a max-conflicts-based heuristic search algorithm [17], namely MCS, specifically for SMTI. Experimental results showed that MCS outperforms both AS and LTIU in terms of execution time and solution quality, making it a more efficient approach for solving SMTI.

3. Proposed algorithm

In this section, we introduce an efficient depth-first search algorithm to solve the SMTI problem. We begin with a detailed description of the algorithm, followed by an illustrative example to demonstrate its execution.

3.1. Algorithm

Our depth-first search algorithm (DFS) for solving the SMTI problem is detailed in Algorithm 1. The input of the algorithm is an instance I of the SMTI problem, represented by two ranking matrices of men and women, denoted by $mr(m_i, w_j)$ and $wr(w_j, m_i)$, where $m_i \in M$ and $w_j \in W$. Since our algorithm is an exhaustive algorithm, it must backtrack to find a maximal stable matching. However, for large SMTI instances, the process can take a long time to find an optimal solution. Therefore, we use a global variable *iter* as a termination condition for the algorithm. Initially, the variable *iter* is set to 0, the matching μ is initialized as an empty set, and the optimal matching μ_{opt} is set to μ (lines 2-4). The algorithm then invokes the function Backtrack(I, m_1, μ, μ_{opt}) with the first man m_1, the current matching μ, and the optimal matching μ_{opt}.

The function Backtrack(I, m_i, μ, μ_{opt}) operates as follows. First, it checks if the current matching μ is stable (line 2). If so, it compares its size to that of the current optimal matching μ_{opt}. If μ is larger than μ_{opt}, it updates μ to μ_{opt} (line 3). Next, the function checks whether all men have been processed (i.e., $m_i = \emptyset$) or if the maximal number of iterations has been reached (line 5). If the former condition is true, it returns the current maximal matching μ_{opt}. If the latter condition is true, it returns an approximate maximal matching μ_{opt}. Otherwise, for each woman w_j ranked in ascending order by man m_i (line 6), if w_j is unmatched or prefers m_i over her current partner m_k (line 8), then w_j is matched to m_i to form a pair (m_i, w_j) in μ (line 9). If w_j has been matched to m_k and rejects m_k, then m_k becomes unmatched (line 10). Next, the function finds some unmatched man m_t (line 11), and call Backtrack(I, m_t, μ, μ_{opt}) for the next process (line 12). Afterward, if μ_{opt} is complete, it immediately returns μ_{opt} (line 13). Otherwise, it undoes all changes to restore the state for backtracking (lines 14-15).

Algorithm 1: Depth-first search algorithm for SMTI problem

```
1.  function DFS(I)
2.      global iter := 0;
3.      μ := ∅;
4.      μ_opt := μ;
5.      return Backtrack(I, m_1, μ, μ_opt);
6.  end
```

```
1.  function μ_opt = Backtrack(I, m_i, μ, μ_opt)
2.      if (Stable(I, μ))
3.          if (|μ_opt| < μ) μ_opt = μ;
4.      end
5.      if (m_i = ∅) or (iter > max_iter) return μ_opt;
```

6. **for** (*each* w_j *ordered in ascending rank by* m_i)
7. $m_k = \mu(w_j)$;
8. **if** (($m_k = \emptyset$) or ($wr(w_j, m_i) < wr(w_j, m_k)$))
9. $\mu := \mu \cup \{(m_i, w_j)\}$;
10. **if** ($m_k \neq \emptyset$) $M(m_k) = 0$;
11. $m_t :=$ *some man who is not matched in* μ;
12. $\mu_{opt} :=$ Backtrack(I, m_t, μ, μ_{opt});
13. **if** ($|\mu_{opt}| = n$) **return** μ_{opt};
14. $\mu := \mu \setminus \{(m_i, w_j)\}$;
15. **if** ($m_k \neq \emptyset$) $\mu := \mu \cup \{(m_k, w_j)\}$;
16. **end**
17. **end**
18. **end**

The function for checking whether a matching μ is stable is shown in Algorithm 2. For each man $m_i \in M$, we let w_j be the partner of m_i in μ (line 3). If m_i is unmatched in μ (i.e., $w_j = \emptyset$), the function skips to the next man (line 4). Otherwise, the function evaluates the stability of μ through two cases:

Case 1 (lines 5-11): For each woman $w_t \in W$, if w_t is already matched to m_i in μ (i.e., $w_t = w_j$) or m_i does not rank w_t in his preference list, the function moves to the next woman (line 6). If m_i prefers w_t over w_j (line 7), the function checks whether w_t is unmatched in μ or prefers m_i over her current partner m_k. If either condition holds, the function returns false (line 9), indicating that μ is unstable.

Case 2 (lines 12-15): For each man $m_k \in M$, if m_k is m_i, the partner of m_k is not w_j, or w_j does not rank m_k in her preference list, the function moves to the next man (line 13). Otherwise, if w_j prefers m_k over her partner m_i, the function returns false (line 14), indicating that μ is unstable.

If the function does not return false in either of two cases above, it returns true (line 17), meaning that the matching μ is stable. It should be noted that the time complexity of this function is $O(n^2)$, since the outer loop iterates $O(n)$ and the inner loop iterates $O(n)$, where n is number of men or women.

Algorithm 2: Check a stable matching

1. **function Stable**(I, μ)
2. **for** (*each* m_i *in* M)
3. $w_j = \mu(m_i)$;
4. **if** ($w_j = \emptyset$) continue;
5. **for** (*each* w_t *in* W)
6. **if** (($w_t = w_j$) or ($mr(m_i, w_t) = 0$) continue;
7. **if** ($mr(m_i, w_t) < mr(m_i, w_j)$)
8. $m_k = \mu(w_t)$;
9. **if** (($m_k = \emptyset$) or ($wr(w_t, m_i) < wr(w_t, m_k)$) **return** false;

10. **end**
11. **end**
12. **for** (*each* m_k *in M*)
13. **if** (($m_k = m_i$) or ($\mu(m_k) \neq w_j$)) or ($wr(w_j, m_k) = 0$) continue;
14. **if** ($wr(w_j, m_k) < wr(w_j, m_i)$) **return** false;
15. **end**
16. **end**
17. **return** true;
18. **end**

3.2. Examples

An illustrative example of an SMTI instance I with six men and six women is shown in Table 1. In the preference lists, ties are indicated using parentheses. For instance, in the men's preference lists, if we write m_1: (w_2 w_3) w_5, it means m_1 ranks w_2, w_3 equally at first, and w_5 second. In this case, we have $mr(m_1, w_2) = mr(m_1, w_3) = 1$, and $mr(m_1, w_5) = 2$. Similarly, in the women's preference lists, if we write w_1: m_4 (m_2 m_3 m_5), it means w_1 ranks m_4 at first, m_2, m_3, and m_5 equally at second. In this case, we have $wr(w_1, m_4) = 1$, and $wr(w_1, m_2) = wr(w_1, m_3) = wr(w_1, m_5) = 2$.

Table 1: *An instance I of the SMTI problem*

Preference lists of men and women		*Rank lists of men and women*	
m_1: (w_2 w_3) w_5	w_1: m_4 (m_2 m_3 m_5)	m_1: 0 1 1 0 2 0	w_1: 0 2 2 1 2 0
m_2: (w_1 w_6) w_2 w_4 w_3	w_2: (m_1 m_2)	m_2: 1 2 4 3 0 1	w_2: 1 1 0 0 0 0
m_3: (w_1 w_3 w_6)	w_3: (m_1 m_2 m_3 m_6)	m_3: 1 0 1 0 0 1	w_3: 1 1 1 0 0 1
m_4: (w_1 w_5)	w_4: m_6 m_2	m_4: 1 0 0 0 1 0	w_4: 0 2 0 0 0 1
m_5: w_6 w_1	w_5: m_6 (m_1 m_4)	m_5: 2 0 0 0 0 1	w_5: 2 0 0 2 0 1
m_6: (w_3 w_4 w_5)	w_6: (m_2 m_3) m_5	m_6: 0 0 1 1 1 0	w_6: 0 1 1 0 2 0

The iteration steps of the algorithm to find a complete matching are shown in Table 2. Initially, the algorithm call Backtrack(I, m_1, μ, μ_{opt}). In step 1, m_1 is unmatched in μ, so he proposes to his most preferred woman w_2 from his list. Since w_2 is unmatched in μ, she is matched to m_1, resulting in $\mu = \{(m_1, w_1)\}$. In step 2, m_2 is unmatched in μ, so he proposes to his most preferred woman $w_1$from his list. Since w_1 is unmatched in μ, she is matched to m_2, yielding $\mu = \{(m_1, w_2), (m_2, w_1)\}$. In step 3, m_3 is unmatched in μ, so he proposes to his most preferred woman w_1 from his list. However, w_1 is already matched to m_2 and prefers m_2 over m_3, leading to m_3 being rejected by w_1. Therefore, the matching remains $\mu = \{(m_1, w_2), (m_2, w_1)\}$. In step 4, m_3 is still unmatched in μ, so he proposes to w_3, his next preferred woman who hasn't yet received a proposal. Since w_3 is unmatched μ, she is matched to m_3, resulting in $\mu = \{(m_1, w_2), (m_2, w_1), (m_3, w_3)\}$. In step 5, m_4 is unmatched in μ, so he proposes to his most preferred woman w_1 from his list. Since w_1 is already matched to m_2 but prefers m_4 over m_2, she rejects m_2 and pairs with m_4, yielding

$\mu = \{(m_1, w_2), (m_3, w_3), (m_4, w_1)\}$. This process repeats until the algorithm achieves a complete matching $\mu = \{(m_1, w_2), (m_2, w_1), (m_3, w_3), (m_4, w_5), (m_5, w_6), (m_6, w_4)\}$.

Table 2: *The iteration steps of the DFS algorithm for the example in Table 1*

Step	m_i	w_j	m_k	μ	m_t
1	m_1	w_2	$\emptyset$	$\{(m_1, w_2)\}$	m_2
2	m_2	w_1	$\emptyset$	$\{(m_1, w_2), (m_2, w_1)\}$	m_3
3	m_3	w_1	m_2	$\{(m_1, w_2), (m_2, w_1)\}$	$\emptyset$
4	m_3	w_3	$\emptyset$	$\{(m_1, w_2), (m_2, w_1), (m_3, w_3)\}$	m_4
5	m_4	w_1	m_2	$\{(m_1, w_2), (m_3, w_3), (m_4, w_1)\}$	m_2
6	m_2	w_1	m_4	$\{(m_1, w_2), (m_3, w_3), (m_4, w_1)\}$	$\emptyset$
7	m_2	w_6	$\emptyset$	$\{(m_1, w_2), (m_2, w_6), (m_3, w_3), (m_4, w_1)\}$	m_5
8	m_5	w_6	m_2	$\{(m_1, w_2), (m_2, w_6), (m_3, w_3), (m_4, w_1)\}$	$\emptyset$
9	m_5	w_1	m_4	$\{(m_1, w_2), (m_2, w_6), (m_3, w_3), (m_4, w_1)\}$	$\emptyset$
10	m_2	$\emptyset$	$\emptyset$	$\{(m_1, w_2), (m_3, w_3), (m_4, w_1)\}$	$\emptyset$
12	m_2	w_2	m_1	$\{(m_1, w_2), (m_3, w_3), (m_4, w_1)\}$	$\emptyset$
13	m_2	w_4	$\emptyset$	$\{(m_1, w_2), (m_2, w_4), (m_3, w_3), (m_4, w_1)\}$	m_5
…	…	…	…	…	…
28	m_6	w_4	$\emptyset$	$\{(m_1, w_2), (m_2, w_1), (m_3, w_3), (m_4, w_5), (m_5, w_6), (m_6, w_4$	$\emptyset$

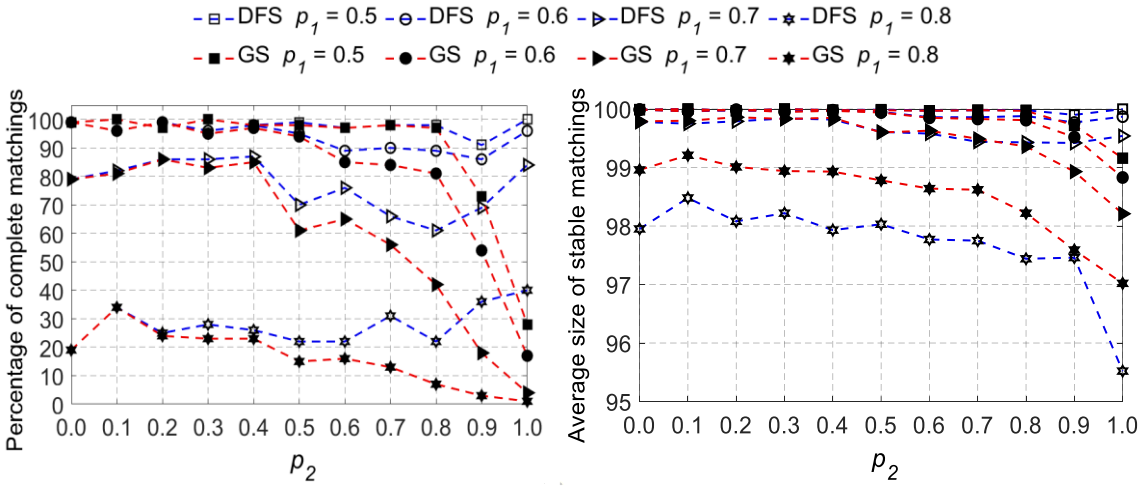
4. Experimental results

This section details the experiments conducted to evaluate the efficiency of the proposed algorithm. Specifically, we compare its performance to the GS algorithm [7] in terms of the matching quality and the execution time. All experiments were carried out using MATLAB 2019a on a personal computer with a Core i7-8550U processor running at 1.8 GHz and 16 GB of RAM.

Datasets: We employed the random problem generator from [11] to create SMTI instances defined by three parameters consisting of n, p_1, and p_2, where n is the size of the instance, p_1 is the probability of incompleteness, and p_2 is the probability of ties. Since stable matchings in these instances consist of acceptable pairs $\{(m_i, w_j) \in M \times W\}$, we set the men's and women's rank matrices such that $mr(m_i, w_j) > 0$ if and only if $wr(w_j, m_i) > 0$, and $mr(m_i, w_j) = 0$ if and only if $wr(w_j, m_i) = 0$. Consequently, each man in an instance ranks approximately $n \times (1 - p_1)$ women, and vice versa. In our experiments, we randomly generated 100 SMTI instances for each combination of parameters (n, p_1, p_2). We then ran the DFS and GS algorithms on these instances, averaging the results to evaluate and compare their efficiency.

Experiment 1: In this experiment, we set $n = 100$, $p_1 \in \{0.1, 0.2, \cdots, 0.8\}$, and $p2 \in \{0.0, 0.1, \cdots, 1.0\}$ to generate SMTI instances. Under this configuration, each man (woman, respectively) ranks approximately 80 women at $p_1 = 0.1$ decreasing to about 10 women (men, respectively) at $p_1 = 0.9$. The maximum number of iterations for the DFS algorithm was set to 20000.

Figure 1 shows the matching quality regarding the percentage of complete matchings and the average size of stable matchings found by the DFS and GS algorithms. We note that when p_1 increases from 0.1 to 0.4, both the DFS and GS algorithms achieve 100% complete matchings, meaning the size of stable matchings in the generated instances is 100. Therefore, we do not include these results in the figure. The experimental results indicate that as p_1 increases from 0.5 to 0.8, the percentage of complete matchings found by DFS and GS decreases. This is because the number of women ranked by men and vice versa decreases, leading to fewer acceptable pairs, making it harder for the DFS and GS algorithms to find complete matchings. However, DFS not only finds a significantly higher percentage of complete matchings compared to GS but also identifies larger stable matchings than GS.



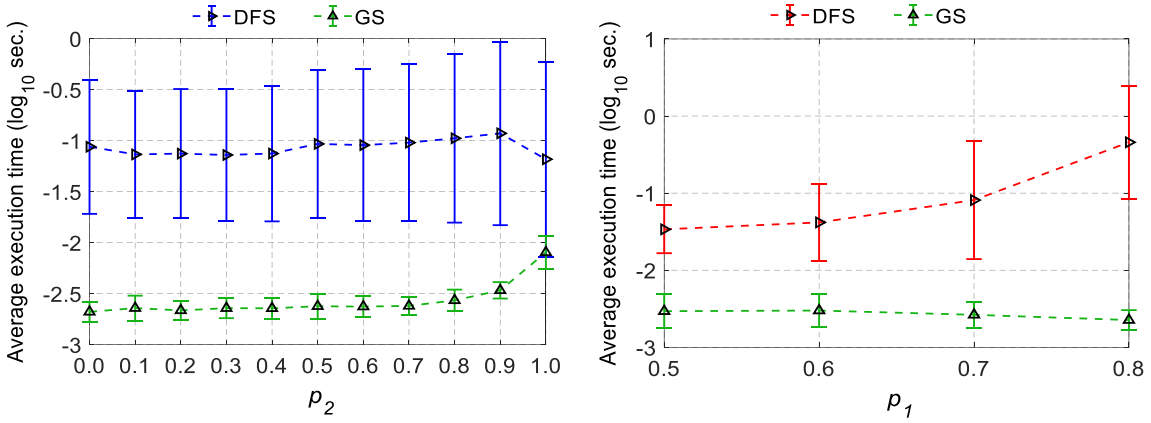
(a) Percentage of complete matchings (b) Average size of stable matchings

Figure 1: *The matching quality regarding the percentage of complete matchings and the average size of stable matchings found by the DFS and GS algorithms. Curves are plotted for p_1 = 0.5, 0.6, 0.7, 0.8 (Experiment 1)*

Figure 2 shows the average execution time over p_1 and p_2 of the DFS and GS algorithms. We observe that as p_2 increases, the average execution time over p_1 for both DFS and GS remains almost unchanged. However, when $p_1$increases, the average execution time over p_2 for DFS increases, while the average execution time over p_2 for GS remains nearly constant. Additionally, the average execution time of DFS is significantly higher than that of GS, indicating that DFS runs much slower than GS.

Experiment 2: In this experiment, we set $n = 200$, $p_1 \in \{0.1, 0.2, \cdots, 0.8\}$, and $p_2 \in \{0.0, 0.1, \cdots, 1.0\}$ to generate SMTI instances. Under this configuration, each man (each woman, respectively) ranks approximately 160 women at $p_1 = 0.1$, decreasing to

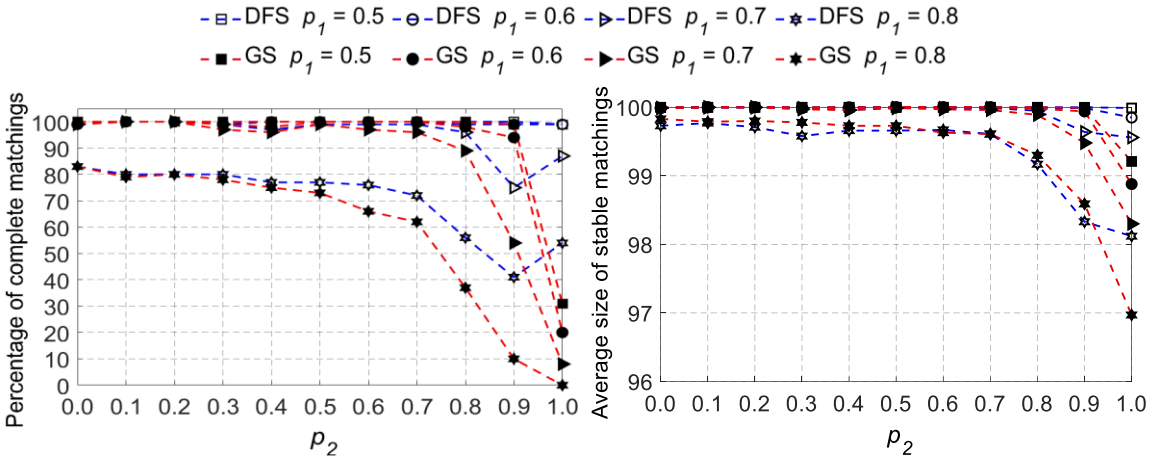
about 20 women (men, respectively) at $p_1 = 0.8$. The maximum number of iterations for the DFS algorithm was set to 40,000.



a) Average execution time over p_1 b) Average execution time over p_2

Figure 2: *The the average execution time over a) p_1 and b) p_2 of the DFS and GS algorithms (Experiment 1)*

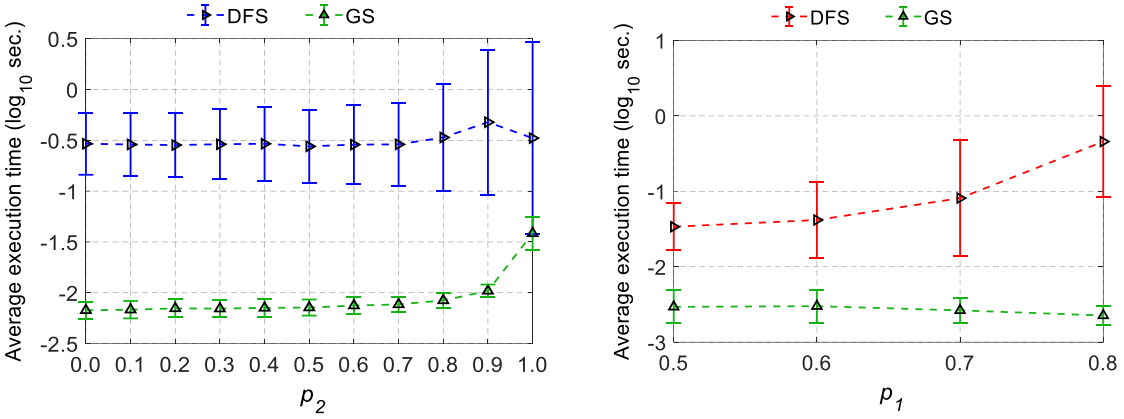
Figure 3 shows the percentage of complete matchings and the average size of stable matchings found by the DFS and GS algorithms. As previously noted, when p_1 increases from 0.1 to 0.4, both DFS and GS achieve 100% complete matchings, with stable matchings reaching the maximum size of 100 in the generated instances. Consequently, these results are not included in the figure. The experimental results reveal that as p_1 increases from 0.5 to 0.8, the percentage of complete matchings found by DFS and GS decreases. Notably, when p_2 increases from 0.8 to 1.0, the matching quality measured by the percentage of complete matchings and the size of stable matchings decreases significantly for GS, while DFS experiences only a slight decrease. This demonstrates that DFS outperforms GS in finding the maximal stable matchings for SMTI instances.



a) Percentage of complete matchings b) Average size of stable matching

Figure 3: *The percentage of complete matchings and the average size of stable matchings found by the DFS and GS algorithms (Experiment 2)*

Figure 4 presents the average execution time over p_1 and p_2 for the DFS and GS algorithms. The results align with those from Experiment 1, showing that the average execution time of DFS is significantly higher than that of GS. This indicates that DFS operates much slower compared to GS.



a) Average execution time over p_1 b) Average execution time over p_2

Figure 4: *The average execution time over a) p_1 and b) p_2 for the DFS and GS algorithms (Experiment 2)*

5. Conclusion

This paper presents a depth-first search algorithm to find maximum stable matchings in the stable marriage problem with ties and incomplete preference lists. Initially, the algorithm sets the matching to an empty set and all man marked as unmatched. In each iteration, an unmatched man proposes to the most preferred woman on his preference list. If the woman is either unmatched or prefers the man over her current partner, she accepts the proposal from the man to form a new pair in the matching. If she rejects her current partner in the matching, the rejected man becomes unmatched, and the algorithm recursively processes the next unmatched man. The algorithm continues until either a complete matching is achieved, or a predefined maximum number of iterations is reached. Experimental results on randomly generated datasets show that our algorithm outperforms the Gale-Shapley algorithm in solution quality but is less efficient in terms of runtime. As an exhaustive algorithm, the proposed algorithm serves as a reliable benchmark for evaluating the solution quality of other approaches to the maximum stable marriage problem with ties and incomplete lists. Future work will focus on optimizing the algorithm's runtime for practical applications.

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. DOI: 10.1080/00029890.1962.11989827

[2] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, "Hard variants of stable marriage," *Theoretical Computer Science*, vol. 276, no. 1-2, pp. 261-279, 2002. DOI: 10.1016/S0304-3975(01)00206-7

[3] M. M. Halldórsson, R.W. Irving, K. Iwama, D. F. Manlove, S. Miyazaki, Y. Morita, and S. Scott, "Approximability results for stable marriage problems with ties," *Theoretical Computer Scien*ce, vol. 306, no. 1-3, pp. 431-447, 2003. DOI: 10.1016/S0304-3975(03)00321-9

[4] K. Iwama, S. Miyazaki, Y. Morita, and D. F. Manlove, "Stable marriage with incomplete lists and ties," In *Proceedings of International Colloquium on Automata, Languages, and Programming*, Prague, Czech Republic, July 11-15, 1999, pp. 443-452. DOI: 10.1007/3-540-48523-6_41

[5] R. W. Irving, "Stable marriage and indifference," *Discrete Applied Mathematics*, vol. 48, no. 3, pp. 261-272, 1994. DOI: 10.1016/0166-218X(92)00179-P

[6] D. Adil, S. Gupta, S. Roy, S. Saurabh, and M. Zehavi, "Parameterized algorithms for stable matching with ties and incomplete lists," *Theoretical Computer Science*, vol. 723, no. 1, pp. 1-10, 2018. DOI: 10.1016/j.tcs.2018.03.015

[7] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh, "Local search for stable marriage problems with ties and incomplete lists," In *Proceedings of 11th Pacific Rim International Conference on Artificial Intelligence*, Daegu, Korea, Aug. 30-Sep. 2, 2010, pp. 64-75. DOI: 10.1007/978-3-642-15246-7_9

[8] Z. Király, "Linear time local approximation algorithm for maximum stable marriage," *Algorithms*, vol. 6, no. 1, pp. 471–484, 2013. DOI: 10.3390/a6030471

[9] D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, and 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, Jan. 25-30, 2015, pp. 1212-1218. DOI: 10.1609/aaai.v29i1.9360

[10] R. W. Irving, D. F. Manlove, and G. O'Malley, "Stable marriage with ties and bounded length preference lists," *Journal of Discrete Algorithms*, vol. 7, no. 1, pp. 213-219, 2009. DOI: 10.1016/j.jda.2008.09.003

[11] 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, July 21-26, 2002, pp. 141-145.

[12] E. McDermid, "A 3/2-approximation algorithm for general stable marriage," In *Proceedings of the 36th International Colloquium on Automata, Languages, and Programming*, Rhodes, Greece, July 5-12, 2009, pp. 689-700. DOI: 10.1007/978-3-642-02927-1_57

[13] Z. Király, "Better and simpler approximation algorithms for the stable marriage problem," *Algorithmica*, vol. 60, no. 1, pp. 3-20, 2011. DOI: 10.1007/s00453-009-9371-7

[14] K. Iwama, S. Miyazaki, and H. Yanagisawa, "A 25/17-approximation algorithm for the stable marriage problem with one-sided ties," *Algorithmica*, vol. 68, no. 1, pp. 758-775, 2014. DOI: 10.1007/s00453-012-9699-2

[15] K. Paluch, "Faster and simpler approximation of stable matchings," *Algorithms*, vol. 7, no. 2, pp. 189-202, 2014. DOI: 10.3390/a7020189

[16] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh, "Local search approaches in stable matching problems," *Algorithms*, vol. 6, no. 4, pp. 591-617, 2013. DOI: 10.3390/a6040591

[17] H. H. Viet, N. T. Uyen, S. W. Lee, T. C. Chung, and L. H. Trang, "A max-conflicts based heuristic search for the stable marriage problem with ties and incomplete lists," *Journal of Heuristics*, vol. 27, no. 1, pp. 439-458, 2021. DOI: 10.1007/s10732-020-09464-8

TÓM TẮT

MỘT THUẬT TOÁN TÌM KIẾM THEO CHIỀU SÂU HIỆU QUẢ CHO VIỆC GIẢI QUYẾT BÀI TOÁN HÔN NHÂN ỔN ĐỊNH TỐI ĐA VỚI DANH SÁCH ƯU TIÊN NGANG BẰNG VÀ KHÔNG ĐẦY ĐỦ

Lê Quốc Anh[1], Đinh Văn Nam[2], Hoàng Hữu Việt[1]
[1]Viện Kỹ thuật và Công nghệ, Trường Đại học Vinh, Nghệ An, Việt Nam
[2]Khoa sư phạm, Trường Đại học Hà Tĩnh, Hà Tĩnh, Việt Nam

Ngày nhận bài 05/10/2024, ngày nhận đăng 06/12/2024

Bài báo này giới thiệu một thuật toán tìm kiếm theo chiều sâu để tìm các phép ghép ổn định tối đa trong bài toán hôn nhân ổn định với danh sách xếp hạng ưu tiên ngang bằng và không đầy đủ. Ý tưởng chính của thuật toán là khởi tạo một phép ghép rỗng, trong đó tất cả người nam đều chưa được ghép cặp. Trong mỗi vòng lặp, một người nam chưa được ghép sẽ chọn một người nữ được xếp hạng thích nhất trong danh sách ưu tiên của mình để ghép cặp. Nếu người nữ chưa được ghép hoặc thích người nam đang chọn mình hơn người nam đang được ghép hiện tại, người nữ sẽ được ghép với người nam đang chọn mình để tạo thành một cặp ghép mới trong phép ghép. Ngược lại, người nữ sẽ từ chối ghép cặp với người nam. Khi một người nam bị từ chối ghép cặp với một người nữ, người này sẽ trở thành người nam chưa được ghép. Thuật toán sẽ tiếp tục xử lý đệ quy cho người nam chưa được ghép tiếp theo cho đến khi tìm thấy một phép ghép hoàn chỉnh hoặc đạt đến một số vòng lặp tối đa cho trước. Kết quả thực nghiệm trên các bộ dữ liệu được tạo ngẫu nhiên chỉ ra rằng thuật toán của chúng tôi hiệu quả về chất lượng nghiệm tìm được của bài toán.

Từ khóa: Tìm kiếm chiều sâu; thuật toán Gale-Shapley; phép ghép ổn định; bài toán hôn nhân ổn định; danh sách ngang bằng và không hoàn chỉnh.