AN EFFICIENT ALGORITHM TO FIND A MAXIMUM WEAKLY STABLE MATCHING FOR SPA-ST PROBLEM

Nguyen Thi Uyen[1]✉, **Tran Xuan Sang**[2]

(1) School of Engineering and Technology, Vinh University, Vietnam
(2) Cyber School, Vinh University, Vietnam
{uyennt,sangtx}@vinhuni.edu.vn

ICAISC International Conference on Artificial Intelligence and Soft Computing Zakopane, June 19-23, 2022

21st International Conference on Artificial Intelligence and Soft Computing Web System

Content

1. Introduction
2. Related works
3. Proposed Algorithm
4. Experiment results
5. Conclusion

Introduction(1) - SPA-ST problem

- The **S**tudent-**P**roject **A**llocation problem with lecturer preferences over **S**tudents containing **T**ies (SPA-ST) is an extension of the Student-Project Allocation problem (SPA).
- The SPA-ST consists of a set of Students ($\mathcal{S}$), a set of Projects ($\mathcal{P}$), and a set of Lecturers ($\mathcal{L}$). Each lecturer has a capacity d_k. Each project is offered by one lecturer and has a capacity c_j. ($c_j, d_k \in \mathbb{Z}^+$).

Table 1. An instance of SPA-ST

Student's preferences	Lecturer's preferences	
s_1: $(p_1\ p_7)$	l_1: $(s_7\ s_4)\ s_1\ s_3\ (s_2\ s_5)\ s_6$	l_1 offers p_1, p_2, p_3
s_2: $p_1\ p_2\ (p_3\ p_4)\ p_5\ p_6$	l_2: $s_3\ (s_2\ s_6)\ s_7\ s_5$	l_2 offers p_4, p_5, p_6
s_3: $(p_2\ p_1)\ p_4$	l_3: $(s_1\ s_7)$	l_3 offers p_7, p_8
s_4: p_2		
s_5: $(p_1\ p_2)\ p_3\ p_4$		
s_6: $(p_2\ p_3)\ p_4\ p_5\ p_6$	Project capacities	$c_1 = 2$, $c_i = 1$, $(2 \leq i \leq 8)$
s_7: $(p_5\ p_3)\ p_8$	Lecturer capacities	$d_1 = 3$, $d_2 = 2$, $d_3 = 2$

Introduction(2) - Definitions

- A matching M of a $\mathrm{SPA\text{-}ST}$ instance is a set of acceptable pairs (s_i, p_j) or $(s_i, \varnothing)$ such that $|M(s_i)| \leq 1$ for all $s_i \in \mathcal{S}$, $|M(p_j)| \leq c_j$ for all $p_j \in \mathcal{P}$, and $|M(l_k)| \leq d_k$ for all $l_k \in \mathcal{L}$.
- A project p_j is *under-subscribed*, *full* or *over-subscribed* according as $|M(p_j)| < c_j$, $|M(p_j)| = c_j$, or $|M(p_j)| > c_j$, respectively.
- A lecturer l_k is *under-subscribed*, *full* or *over-subscribed* according as $|M(l_k)| < d_k$, $|M(l_k)| = d_k$, or $|M(l_k)| > d_k$, respectively.
- If $(s_i, p_j) \in M$, then s_i is matched to p_j, denoted by $M(s_i) = p_j$. If $M(s_i) = \varnothing$, then s_i is *unmatched* in M.
- Example
  $M = \{(s_1, p_1), (s_2, \varnothing), (s_3, p_4), (s_4, p_2), (s_5, \varnothing), (s_6, p_5), (s_7, p_3)\}$, where s_2, s_5 are unmatched, p_1 is *under-subscribed*, and l_1 is *full*.

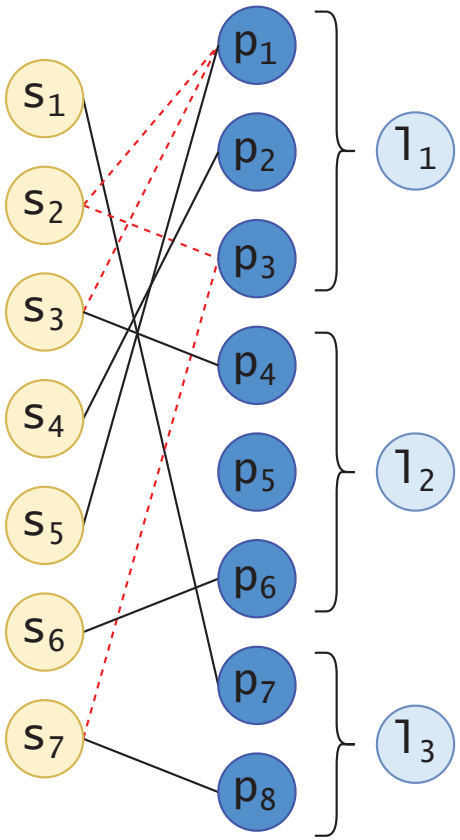
Introduction(3) - Definitions

Let $(s_i, p_j) \in (\mathcal{S} \times \mathcal{P}) \setminus M$ be a blocking pair for a weakly stable matching M if the following conditions are satisfied:

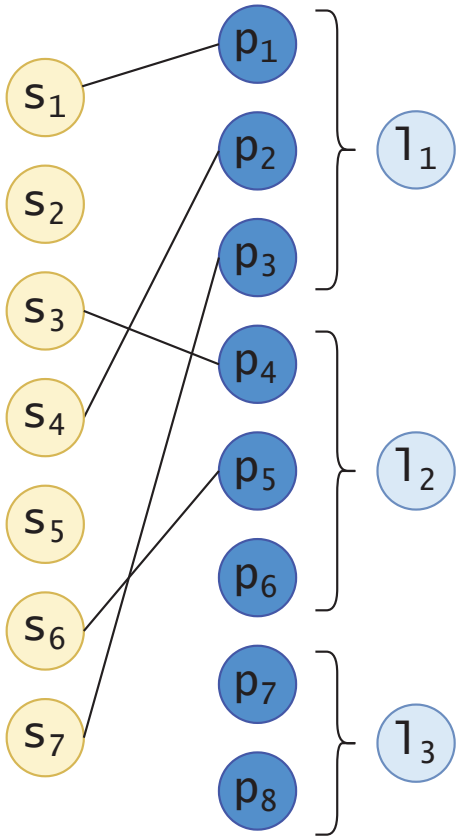
1. s_i and p_j find accept each other;
2. s_i prefers p_j to $M(s_i)$ or $M(s_i) = \varnothing$;
3. either (a), (b) or (c) holds as follows:
   - $|M(p_j)| < c_j$ and $|M(l_k)| < d_k$;
   - $|M(p_j)| < c_j$, $|M(l_k)| = d_k$ and;
     - either $s_i \in M(l_k)$ or;
     - l_k prefers s_i to the worst student in $M(l_k)$;
   - $|M(p_j)| = c_j$ and l_k prefers s_i to the worst student in $M(p_j)$.

- A matching M is called *weakly stable* if it admits no blocking pair, otherwise it is called *unstable*. $|M|$ is called the number of matched students in M. If $|M| =$ n, then M is a *perfect* matching, otherwise, M is a *non-perfect* matching.

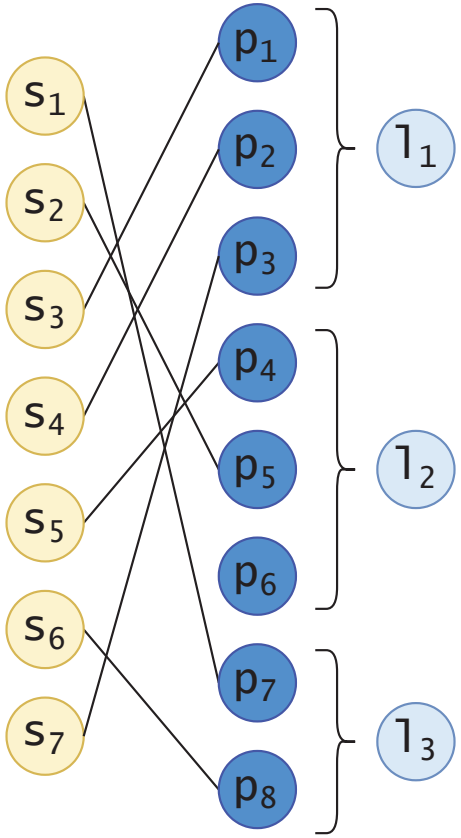
Introduction(4) - Definitions



M is unstable



M is stable



M is perfect

Related works

- Several researchers have focused on solving the $\mathrm{SPA\text{-}ST}$ problem because of its applications at the Glasgow University [Kwanashie et al.2014], Southern Denmark University [Chiarandini et al.2019], York University [Kazakov et al.2002].
- Cooper et al.2018 presented a $3/2$- approximation algorithm, called AP, to find a *weakly stable* matching
- Besides, they also modeled the $\mathrm{SPA\text{-}ST}$ problem as an Integer Programming (IP) problem.
- Olaosebikan et al.2020 described the polynomial-time algorithm to find a *strongly stable* matching.
- Olaosebikan et al.2020 proposed an approximation algorithm for solving $\mathrm{SPA\text{-}ST}$ problem in terms of finding a *super-stable* matching.

Proposed Algorithm(1) - Main idea

- This paper presents an effective heuristic algorithm to solve the $\mathrm{MAX\text{-}SPA\text{-}ST}$ problem of large sizes.
- Our main idea is to start from a stable matching, then define two heuristic strategies promoting *unmatched students* and *under-subscribed* lecturers to improve the matching size by breaking stable pairs.
- Our algorithm terminates when it finds a *perfect matching* or reaches a maximum number of iterations.
- The experimental results show that our proposed algorithm is more efficient than the AP (Cooper et al.2018) algorithm in terms of solution quality and execution time.

Proposed Algorithm(2) - Pseudocode

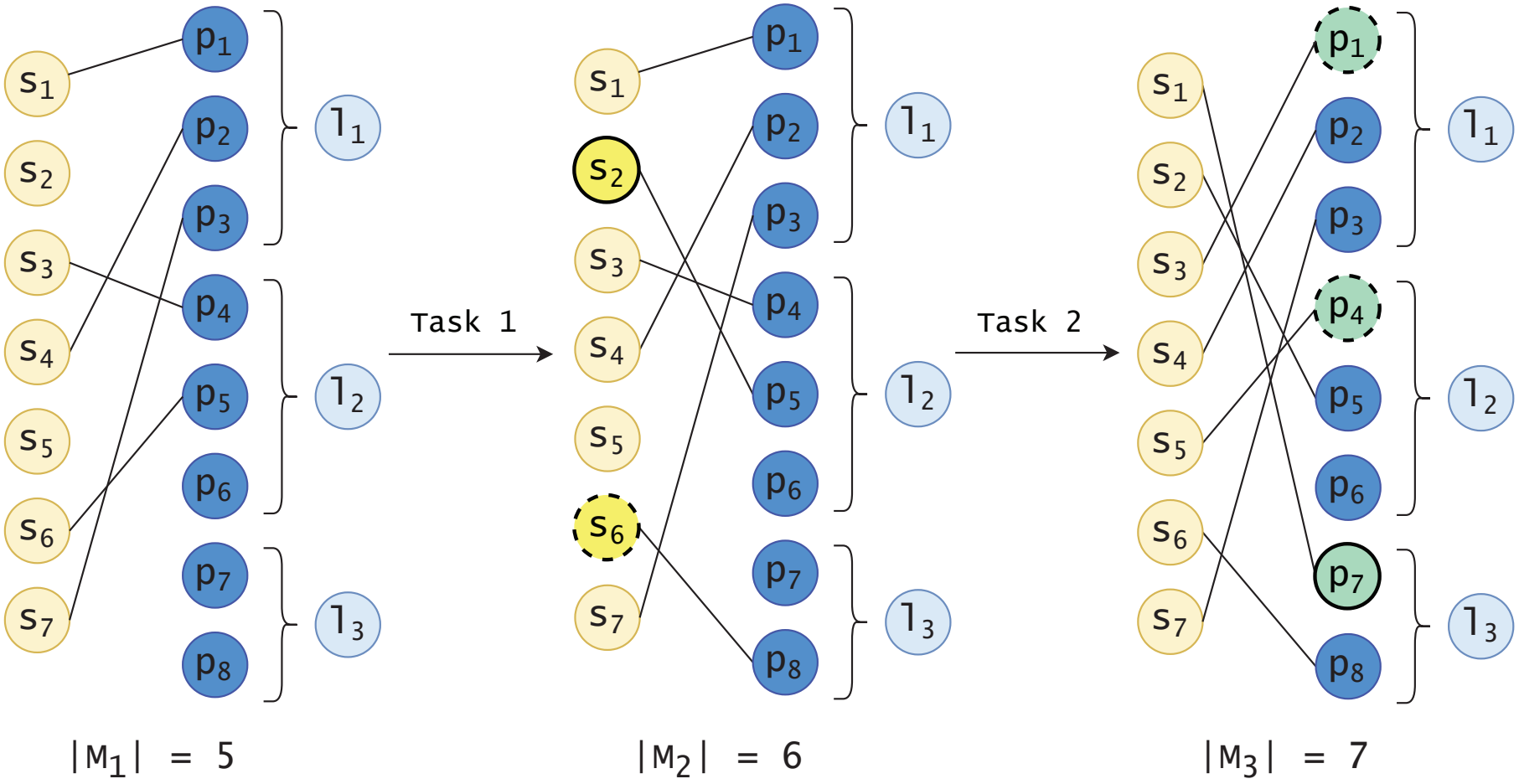
Algorithm 1: HA Algorithm for MAX-SPA-ST problem

Input: - An SPA-ST instance I.
- max_iter is the maximum number of iterations.

Output: A maximum stable matching M.

```
function HA(I)
    M := EGS(I);                                   ▷ generate a stable matching
    v(s_i) := 0, (1 ≤ i ≤ n);                      ▷ mark the replacing time of s_i
    v(p_i) := 0, (1 ≤ i ≤ q);                      ▷ mark the replacing time of p_i
    iter := 0;
    while (iter ≤ max_iter) do
        iter := iter + 1;
        if |M| = n then break ;
        s_i := a random unmatched student in M;                    ▷ Task 1
        R'_{s_i} := s_i's ranks list;
        while R'_{s_i} is non-empty do
            p_j := argmin(R'_{s_i} > 0), ∀p_j ∈ P;
            l_k := a lecturer who offers p_j;
            for (each s_t ∈ M(l_k)| R_{l_k}(s_t) = R_{l_k}(s_i)) do
                if (|M(p_j)| < c_j) or (s_t ∈ M(p_j) and |M(p_j)| = c_j) then
                    if v(s_i) ≥ v(s_t) or a small probability then
                        M := M \ {(s_t, p_z)} ∪ {(s_i, p_j)}| p_z = M(s_t);
                        v(s_i) := v(s_i) + 1;
                        Repair(p_z, l_k);
                        M := Break_Student(M, s_t);
                        break;
            if M(s_i) ≠ ∅ then break;
            else R'_{s_i}(p_j) := 0 ;
        p_i := a random under-subscribed project in M;             ▷ Task 2
        l_k := a lecturer who offers p_i;
        for each s_j ∈ S|R_{s_j}(p_t) = R_{s_j}(p_i)|p_t = M(s_j) do
            if (|M(l_k)| < d_k) or (|M(l_k)| = d_k and s_j ∈ M(l_k)) then
                if (v(p_i) ≥ v(p_t) or a small probability then
                    M := M \ {(s_j, p_t)} ∪ {(s_j, p_i)};
                    v(p_i) := v(p_i) + 1;
                    M := Break_Lecturer(M, p_t);
                    break;
    return M;
end function
```

Proposed Algorithm(3) - Example



Experiment results

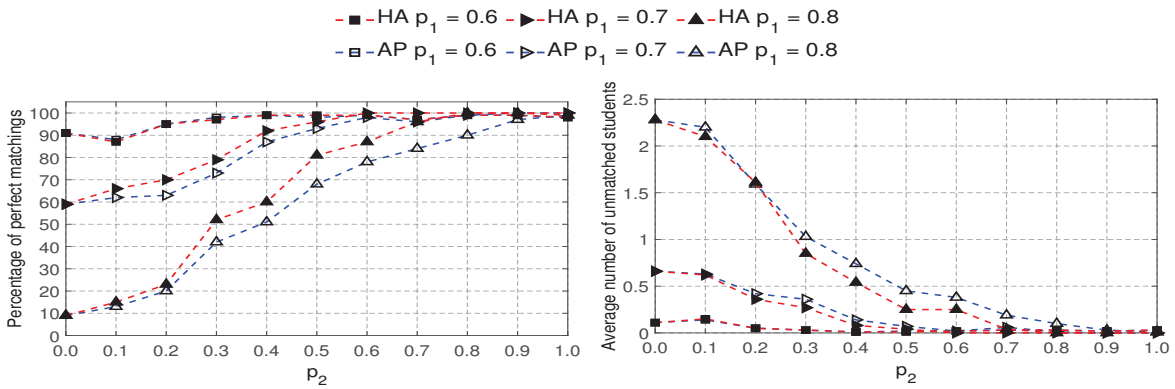
Dataset

- In this section, we compared the solution quality and execution time of HA with those of AP which is an approximation algorithm (Cooper at al.2018) for the $\mathrm{MAX\text{-}SPA\text{-}ST}$ problem. We implemented these algorithms by Matlab R2019a software on a Xeon-R Gold 6130 CPU 2.1 GHz computer with 16 GB RAM.
- To perform the experiments, we generated randomly $\mathrm{SPA\text{-}ST}$ instances with five parameters (n, m, q, p_1, p_2), where n is the number of students, m is the number of lecturers, q is the number of projects, p_1 is the probability of incompleteness, and p_2 is the probability of ties. By this setting, on average, each student ranks about $q \times (1 - p_1)$ projects. In our experiments, we set the total capacity of projects and lecturers as $C = 1.2n$ and $D = 1.1n$, respectively.

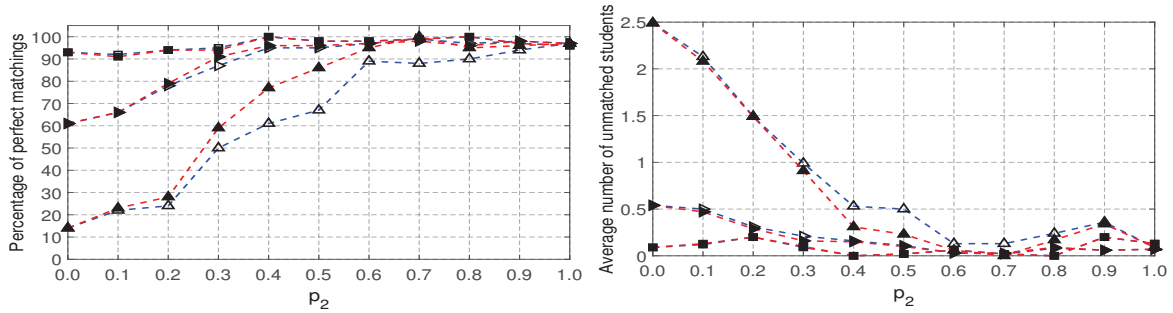
Experiment results

Comparison of solution quality

- **Experiment 1**: Firstly, we randomly generated 100 instances of SPA-ST for parameters (n, m, q, p_1, p_2) with $n \in \{100, 200\}$, $m = 0.05n$, $q = 0.1n$, $p_1 \in [0.1, 0.8]$ with step 0.1, and $p_2 \in [0.0, 1.0]$ with step 0.1.



(a) – Solution quality for $n = 100$, $m = 5$, $q = 10$ – (b)

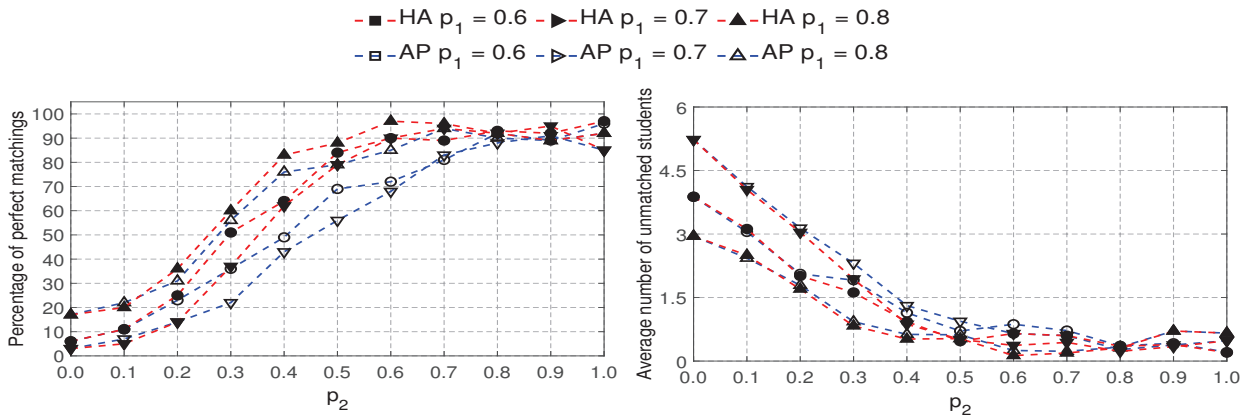


(c) – Solution quality for $n = 200$, $m = 10$, $q = 20$ – (d)

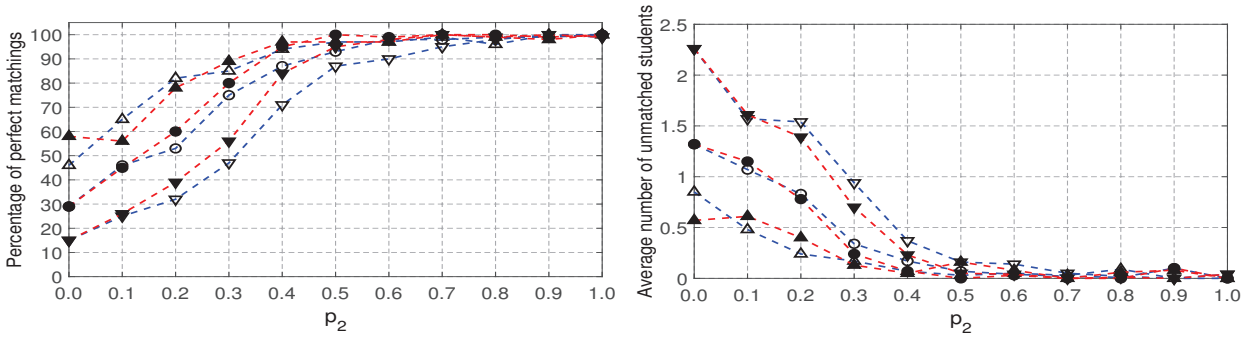
Experiment results

Comparison of solution quality

- **Experiment 2**: In this experiment, we changed $n \in \{300, 400\}$, $p_1 \in \{0.82, 0.84, 0.86\}$ and kept the values of m, q, and p_2 as in Experiment 1



(a) – Solution quality for $n = 300$, $m = 15$, $q = 30$ – (b)



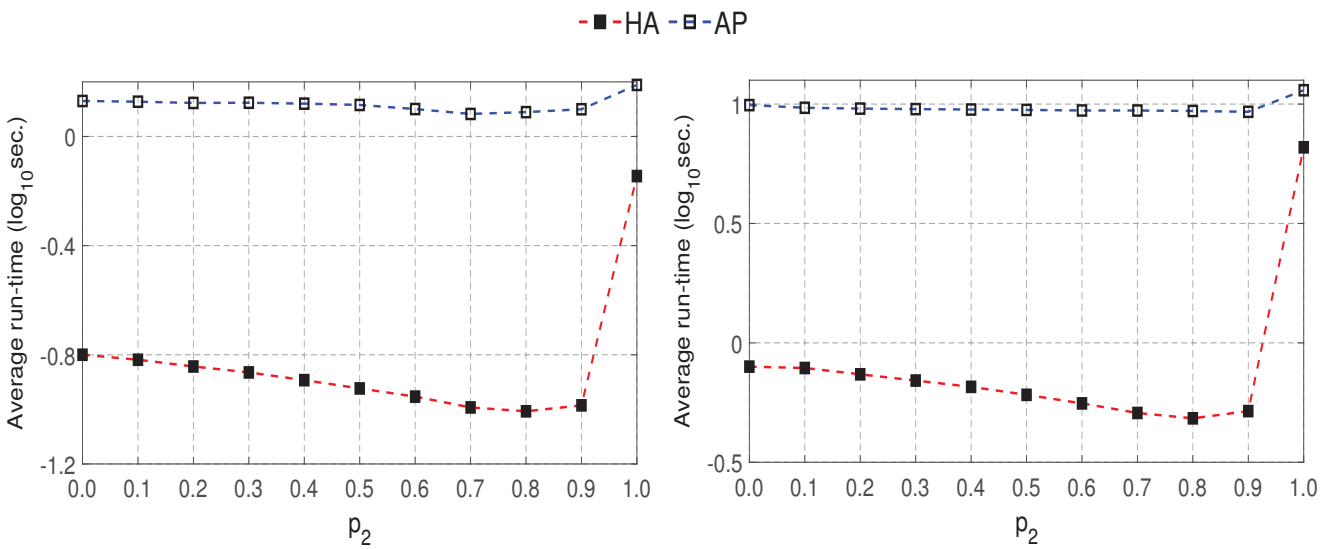
(c) – Solution quality for $n = 400$, $m = 20$, $q = 40$ – (d)

Fig. 2. Percentage of perfect matching and average number of unmatched students

Experiment results

Comparison of execution time

- **Experiment 3**: We randomly generated 100 instances of SPA-ST for parameters (n, m, q, p_1, p_2) with $n \in \{1000, 2000\}$, $m = 0.05n$, $q = 0.4n$, $p_1 \in [0.1, 0.8]$ with step 0.1, and $p_2 \in [0.0, 1.0]$ with step 0.1

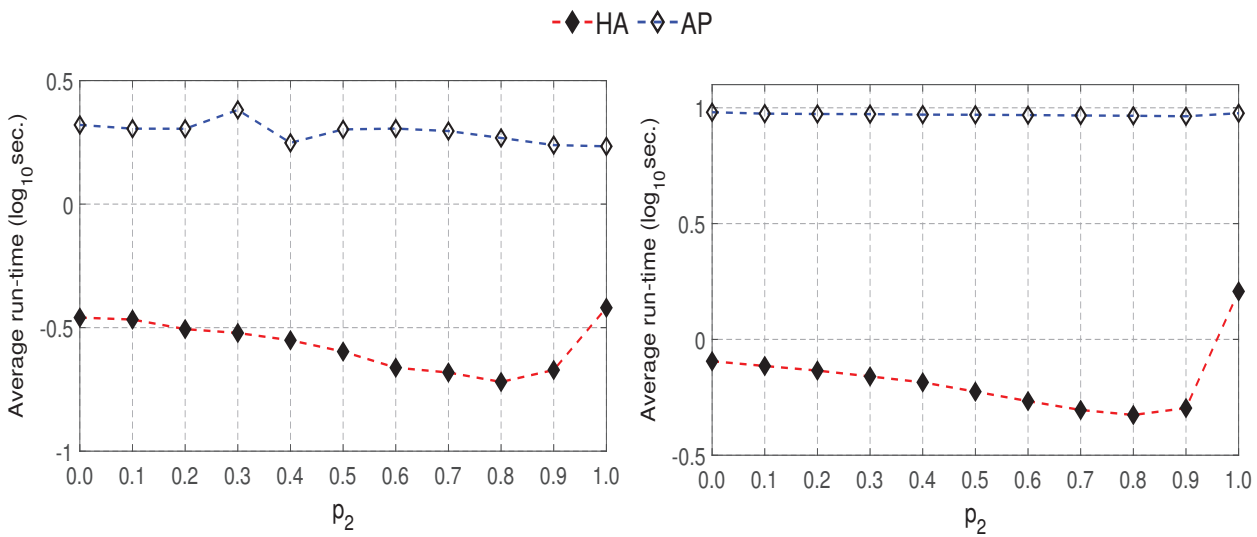


(a) $n = 1000, p_1 \in [0.1, 0.8]$ (b) $n = 2000, p_1 \in [0.1, 0.8]$

Experiment results

Comparison of execution time

- **Experiment 4**: Finally, we kept the values of n, m, q, and p_2 as in Experiment 3, increased the values of $p_1 \in [0.81, 0.89]$ with step 0.01, and randomly generated 100 instances of $\mathrm{SPA\text{-}ST}$ for each combination of values (p_1, p_2). By increasing the values of p_1, we aim to reduce the number of projects ranked by each student compared to Experiment 3.



(c) $n = 1000, p_1 \in [0.81, 0.89]$ (d) $n = 2000, p_1 \in [0.81, 0.89]$

Conclusion

- In this study, we presented a heuristic algorithm for solving the $\mathrm{MAX\text{-}SPA\text{-}ST}$ problem.
- We started with a stable matching and improved the matching size by defining two heuristic strategies to pair the *unmatched* students and *under-subscribed* projects.
- The experimental results showed that our proposed algorithm is efficient in terms of solution quality and execution time for the $\mathrm{MAX\text{-}SPA\text{-}ST}$ problem of large sizes.
- In the future, we will extend this proposed approach to solve the other variants of the SPA problem.

Thank for listening!

ICAISC International Conference on Artificial Intelligence and Soft Computing Zakopane, June 19-23, 2022

21st International Conference on Artificial Intelligence and Soft Computing Web System

Lecture Notes in Computer Science LNCS LNAI LNBI

Springer the language of science

Committee on Informatics of the Polish Academy of Sciences

PAN COMMITTEE ON INFORMATICS POLISH ACADEMY OF SCIENCES

Poland Section of IEEE Computer Society Chapter C16

IEEE computer society

IEEE Poland Section Computational Intelligence Society Chapter