Improved Approximation Bounds for the Student-Project Allocation Problem with Preferences over Projects☆

Kazuo Iwama[a], Shuichi Miyazaki[b,*], Hiroki Yanagisawa[c]

[a] *Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku Kyoto-shi, Kyoto 606-8501, Japan*

[b] *Academic Center for Computing and Media Studies, Kyoto University, Yoshida-Honmachi, Sakyo-ku Kyoto-shi, Kyoto 606-8501, Japan*

[c] *IBM Research - Tokyo, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa 242-8502, Japan.*

Abstract

Manlove and O'Malley [8] proposed the Student-Project Allocation Problem with Preferences over Projects (SPA-P). They proved that the problem of finding a maximum stable matching in SPA-P is APX-hard and gave a polynomial-time 2-approximation algorithm. In this paper, we give an improved upper bound of 1.5 and a lower bound of 21/19 (> 1.1052).

Keywords: The student project allocation problem, Stable matching, NP-hardness, Approximation algorithm, Approximation ratio

---

1. Introduction

Assignment problems based on the preferences of participants, which originated from the famous *Hospitals/Residents problem* (*HR*) [3], are important

☆A preliminary version of this paper was presented at the 8th Annual Conference on Theory and Applications of Models of Computation, TAMC 2011. This work was supported by KAKENHI 22240001 and 20700009.

*Corresponding author

Email addresses: iwama@kuis.kyoto-u.ac.jp (Kazuo Iwama), shuichi@media.kyoto-u.ac.jp (Shuichi Miyazaki), yanagis@jp.ibm.com (Hiroki Yanagisawa)

almost everywhere, such as in education systems where students must be allocated to elementary schools or university students to projects. In the university case, each student may have preferences over certain research projects supervised by professors and usually there is an upper bound on the number of students each project can accept. Our basic goal is to find a "stable" allocation where no students (or projects or professors if they also have preferences over students) can complain of unfairness. This notion of stability was first introduced by Gale and Shapley in the context of the famous *Stable Marriage problem* in 1962 [2].

The *Student-Project Allocation problem* (*SPA*) is a typical formulation of this kind of problem originally described by Abraham, Irving, and Manlove [1]. The participants here are *students*, *projects*, and *lecturers*. Each project is offered by a single lecturer, though one lecturer may offer multiple projects. Each project and each lecturer has a *capacity*. Students have preferences over projects, and lecturers have preferences over students. Our goal is to find a stable matching between students and projects satisfying all of the capacity constraints for projects and lecturers. They proved that all stable matchings for a single instance have the same size, and proposed linear-time algorithms to find one [1].

Manlove and O'Malley [8] proposed a variant of SPA, called *SPA with Preferences over Projects* (*SPA-P*), where lecturers have preferences over projects they offer rather than preferences over students. In contrast to SPA, they pointed out that the sizes of stable matchings may differ, and proved that the problem of finding a maximum stable matching in SPA-P, denoted *MAX-SPA-P*, is APX-hard. They also presented a polynomial-time 2-approximation algorithm. Specifically, they provided a polynomial-time algorithm that finds a stable matching, and proved that any two stable matchings differ in size by at most a factor of two.

Our Contributions. In this paper, we improve both the upper and lower bounds on the approximation ratio for MAX-SPA-P. We give an upper bound of 1.5 and a lower bound of 21/19 (> 1.1052) (under the condition that P $\neq$ NP). For the upper bound, we modify Manlove and O'Malley's algorithm SPA-P-APPROX [8] using Király's idea [7] for the approximation algorithm to find a maximum stable matching in a variant of the stable marriage problem (*MAX-SMTI*). We also show that our analysis is tight. For the lower bound, we give a gap-preserving reduction from (a variant of) MAX-SMTI. Our reduction also gives a lower bound of 1.25 under the Unique Games

Conjecture.

2. Preliminaries

Here we give a formal definition of SPA-P and MAX-SPA-P, derived directly from the literature [8]. An instance I of SPA-P consists of a set S of *students*, a set P of *projects*, and a set L of *lecturers*. Each lecturer $\ell_k \in L$ offers a subset P_k of projects. Each project is offered by exactly one lecturer, i.e., $P_{k_1} \cap P_{k_2} = \emptyset$ if $k_1 \neq k_2$. Each student $s_i \in S$ has an *acceptable* set of projects, denoted A_i, and has a strict order on A_i according to preferences. Each lecturer ℓ_k also has a strict order on P_k according to preferences. Also, each project p_j and each lecturer ℓ_k has a positive integer, called a *capacity*, denoted c_j and d_k, respectively.

An *assignment* M is a subset of $S \times P$ where $(s_i, p_j) \in M$ implies $p_j \in A_i$. Let $(s_i, p_j) \in M$ and ℓ_k be the lecturer who offers p_j. Then we say that s_i *is assigned to* p_j in M, and p_j *is assigned* s_i in M. We also say that s_i *is assigned to* ℓ_k in M and ℓ_k *is assigned* s_i in M.

For $s \in S$, let $M(s)$ be the set of projects to which s is assigned in M. For $r \in P \cup L$, let $M(r)$ be the set of students assigned to r in M. If $M(s_i) = \emptyset$, we say that the student s_i is *unassigned* in M, otherwise s_i is *assigned* in M. We say that the project p_j is *under-subscribed*, *full*, or *over-subscribed* with respect to M according to whether $|M(p_j)| < c_j$, $|M(p_j)| = c_j$, or $|M(p_j)| > c_j$, respectively, under M. If $|M(p_j)| > 0$, we say that p_j is *non-empty*, otherwise, it is *empty*. Corresponding definitions apply to each lecturer ℓ.

A *matching* M is an assignment such that $|M(s_i)| \leq 1$ for each s_i, $|M(p_j)| \leq c_j$ for each p_j, and $|M(\ell_k)| \leq d_k$ for each ℓ_k. For a matching M, if $|M(s_i)| = 1$, we may use $M(s_i)$ to denote the unique project to which s_i is assigned. The *size* of a matching M, denoted $|M|$, is the number of students assigned in M.

Given a matching M, a (student, project) pair (s_i, p_j) *blocks* M, or is a *blocking pair* for M, if the following three conditions are met:

1. $p_j \in A_i$.
2. Either s_i is unassigned or s_i prefers p_j to $M(s_i)$.
3. p_j is under-subscribed and either
   (a) $s_i \in M(\ell_k)$ and ℓ_k prefers p_j to $M(s_i)$, or
   (b) $s_i \notin M(\ell_k)$ and ℓ_k is under-subscribed, or
   (c) $s_i \notin M(\ell_k)$, ℓ_k is full, and ℓ_k prefers p_j to ℓ_k's worst non-empty

project,

where ℓ_k is the lecturer who offers p_j.

Given a matching M, a *coalition* is a set of students $\{s_{i_0}, s_{i_1}, \ldots, s_{i_{r-1}}\}$ for some $r \geq 2$ such that each s_{i_j} is assigned in M and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where $j+1$ is taken modulo r. A matching that has no blocking pair nor coalition is *stable*. Refer to [8] for the validity of this definition of stability. SPA-P is the problem of finding a stable matching, and MAX-SPA-P is the problem of finding a maximum stable matching.

We say that A is an r-approximation algorithm if it satisfies $OPT(I)/A(I) \leq r$ for all instances I, where $OPT(I)$ and $A(I)$ are the sizes of the optimal and the algorithm's solutions for I, respectively.

3. Approximability

3.1. Algorithm SPA-P-APPROX-PROMOTION

Manlove and O'Malley's algorithm SPA-P-APPROX [8] proceeds as follows. First, all students are unassigned. Any student (s) who has non-empty preference list applies to the top project (p) on the current list of s. If the lecturer (ℓ) who offers p has no incentive to accept s for p, then s is rejected. When rejected, s deletes p from the list. Otherwise, (s, p) is added to the current matching. If, as a result, ℓ becomes over-subscribed, ℓ rejects a student from ℓ's worst non-empty project to satisfy the capacity constraint. This continues until there is no unassigned student whose preference list is non-empty. Manlove and O'Malley proved that the obtained matching is stable.

We extend SPA-P-APPROX using Király's idea [7]. During the execution of our algorithm SPA-P-APPROX-PROMOTION, each student has one of two states, "unpromoted" or "promoted". At the beginning, all of the students are unpromoted. The application sequence is unchanged. When a student (s) becomes unassigned with her preference list exhausted, s is promoted. When promoted, s returns to her original preference list (i.e., all of the previous deletions are canceled) and starts a second sequence of applications from the top of her list. For the decision rule for acceptance or rejection by the lecturers, they will prefer promoted students to unpromoted students within the same project. The formal description of SPA-P-APPROX-PROMOTION is given as Algorithm 1.

When a student s_i applies to a project p_j but is instantly rejected, we say that p_j *rejects* s_i and s_i *is rejected by* p_j. Similarly, when a student s_i being assigned to p_j is rejected (due to another student's application), we say that p_j *rejects* s_i and s_i *is rejected by* p_j.

3.2. Correctness

It is straightforward to show that SPA-P-APPROX-PROMOTION outputs a matching in polynomial time. We will now show that the output matching M is stable. We first prove two useful lemmas:

Lemma 3.1. *Suppose that, during the execution of* SPA-P-APPROX-PROMOTION, *a project p_a rejected a promoted student. Then (i) after that point, no student can be accepted to p_a, and (ii) no unpromoted student can be assigned to p_a in M.*

PROOF. Suppose that a promoted student s is rejected by p_a. Let ℓ_k be the lecturer who offers p_a. It is easy to see that just after this rejection, no unpromoted student can be assigned to p_a. We show that after that point, if a student s' applies to p_a when there is no unpromoted student assigned to p_a, then s' must be rejected. It is easy to see that the lemma follows by using this fact inductively.

Note that just after this rejection, either (1) p_a is full or (2) p_a is under-subscribed and ℓ_k is full. We consider Case (2) first. Since p_a is under-subscribed but s was rejected by p_a, just before this rejection p_a must be ℓ_k's worst non-empty project or even worse than ℓ_k's worst non-empty project. Then after this rejection, p_a remains ℓ_k's worst non-empty project or worse than that. Note that now ℓ_k remains full until the end of the execution. Then after this point, when any student applies to p_a, only Cases A (line 10) or B (line 16) of the algorithm can apply. Since there is no unpromoted student in $M(p_a)$, s' must be rejected.

In Case (1), if p_a is still full when s' applies to p_a, Case A of the algorithm applies and hence s' must be rejected since $M(p_a)$ contains no unpromoted student. If p_a is under-subscribed when s' applies to p_a, then some student was already rejected by p_a. At that time, ℓ_k must have been full and p_a was ℓ_k's worst non-empty project. Therefore, ℓ_k is still full and p_a is ℓ_k's worst non-empty project or worse than ℓ_k's worst non-empty project. Then we can apply the same argument as in Case (2). □

Lemma 3.2. *Suppose that, during the execution of* SPA-P-APPROX-PROMOTION, *a project p_a has rejected a student. Then after that point, no unpromoted student can be accepted to p_a.*

Proof. The proof is basically similar to that of the previous lemma, and hence we give only a brief sketch. Let ℓ_k be the lecturer who offers p_a. After the rejection point, ℓ_k or p_a is full. If ℓ_k is full but p_a is under-subscribed, then p_a must be ℓ_k's worst non-empty project or worse than that. Then, afterwards, ℓ_k has no incentive to accept an unpromoted student to p_a. Next, suppose that p_a is full after the rejection point. As long as p_a remains full, p_a rejects an unpromoted student. If p_a becomes under-subscribed, then ℓ_k must be full and p_a is ℓ_k's worst non-empty project or worse than that. Hence we can apply the same argument as the former case. □

To prove the stability, we need to prove that there is no coalition or blocking pair.

Lemma 3.3. *The output matching M is coalition-free.*

Proof. Suppose that there is a coalition $\{s_{i_0}, s_{i_1}, \ldots, s_{i_{r-1}}\}$ for some $r \geq 2$. Let $p_{i_j} = M(s_{i_j})$ for each j ($0 \leq j \leq r-1$). Thus s_{i_j} prefers $p_{i_{j+1}}$ to p_{i_j} (where $j+1$ is taken modulo r). Therefore, at some point of the execution, $p_{i_{j+1}}$ was deleted from s_{i_j}'s list. Note that during the execution of the algorithm, one project may be deleted from a student's list twice (because of a promotion). Hereafter, a "deletion" means the final deletion unless otherwise stated.

Now suppose without loss of generality that among such deletions, the first occurrence was the deletion of p_{i_1} from s_{i_0}'s list. First, suppose that s_{i_0} is eventually unpromoted. Note that s_{i_1} applied to and was accepted by p_{i_1} after s_{i_0} was rejected by p_{i_1}. Therefore s_{i_1} is eventually promoted by Lemma 3.2. Then s_{i_1} was rejected by p_{i_2} when s_{i_1} was promoted. This means that s_{i_2} is eventually promoted by Lemma 3.1(ii). Repeating this argument, we can conclude that $s_{i_{r-1}}$ is eventually promoted. Then this contradicts Lemma 3.1(ii) since p_{i_0} rejected the promoted student $s_{i_{r-1}}$ but is assigned an unpromoted student s_{i_0} in M.

Next suppose that s_{i_0} is eventually promoted. Then since p_{i_1} rejected a promoted student s_{i_0}, after that p_{i_1} accepts no student by Lemma 3.1(i). This contradicts the fact that s_{i_1} was accepted to p_{i_1} later. □

Lemma 3.4. *The output matching M has no blocking pair.*

Proof. Assume that there exists a blocking pair (s_r, p_t) for M. Then it is clear that s_r was rejected by p_t during the execution (recall that this rejection is the second one if s_r was eventually promoted). Let ℓ_k be the lecturer who offers p_t. Rejections occur at lines 12, 14, 17, 23, and 25. If this rejection occurred at line 17, 23, or 25, then p_t was already ℓ_k's worst non-empty project or worse than that, and this is also the case in M. We know that ℓ_k was full at this rejection point, and remains full in M. Therefore, (s_r, p_t) cannot block M. If this rejection occurred at line 12 or 14 as a result of ℓ_k being full and p_t being ℓ_k's worst non-empty project, then the same argument holds. Therefore suppose that this rejection occurred at line 12 or 14 as a result of p_t being full. Since (s_r, p_t) blocks M, p_t is under-subscribed in M. Then p_t changed from being full to being under-subscribed at some point. This can happen only when ℓ_k is full and p_t is ℓ_k's worst non-empty project. Again, we can use the same argument to show that (s_r, p_t) cannot block M, a contradiction. $\square$

The following lemma follows immediately from Lemmas 3.3 and 3.4.

Lemma 3.5. SPA-P-APPROX-PROMOTION *returns a stable matching.*

3.3. Analysis of the Approximation Ratio

For a given instance I, let M be a matching output from SPA-P-APPROX-PROMOTION, and let M_{opt} be a largest stable matching for I.

Lemma 3.6. $|M_{opt}| \leq \frac{3}{2}|M|$.

PROOF. Based on M and M_{opt}, we define a bipartite graph $G_{M,M_{opt}} = (U, V, E)$ as follows: Each vertex in U corresponds to a student in I, and each vertex in V corresponds to a position of a project in I. Precisely speaking, for each project p_j whose capacity is c_j, we create c_j "positions" of p_j, each of which can accept at most one student, and each vertex in V corresponds to each such position. We use s_i to denote the vertex in U corresponding to a student s_i and $p_{j,1}, p_{j,2}, \ldots, p_{j,c_j}$ to denote the vertices in V corresponding to a project p_j.

If a student s_i is assigned to a project p_j in M (M_{opt}, respectively), we include an edge $(s_i, p_{j,t})$ for some t ($1 \leq t \leq c_j$), called an M-edge (M_{opt}-edge, respectively), in E. If s_i is assigned to the same project p_j both in M and M_{opt}, then M- and M_{opt}-edges corresponding to this assignment include the same position of p_j, which means we give parallel edges $(s_i, p_{j,t})$ for some t. We also ensure that there are no two vertices p_{j,t_1} and p_{j,t_2} such that p_{j,t_1} is matched in M but not in M_{opt}, and p_{j,t_2} is matched in M_{opt} but not in M. In such a case, there will be M-edge (s_{i_1}, p_{j,t_1}) and M_{opt}-edge (s_{i_2}, p_{j,t_2}). Then we can remove (s_{i_1}, p_{j,t_1}) and add (s_{i_1}, p_{j,t_2}) instead.

Note that each vertex of $G_{M,M_{opt}}$ has degree at most two. Therefore its connected components (other than isolated vertices) are alternating paths or alternating cycles. Now we will modify $G_{M,M_{opt}}$ while retaining this property and keeping the numbers of M-edges and M_{opt}-edges unchanged. Note that the resulting graph may not correspond to a feasible solution for I. We use this modification only for the purpose of comparing the sizes of M and M_{opt}.

A connected component consisting of only one M_{opt}-edge is called a *Type-I component*. A connected component which is a length-three alternating path consisting of two M_{opt}-edges and one M-edge in the middle is called a *Type-II component*. We show that there are no Type-I or Type-II components in the resulting bipartite graph. If this is true, the connected component having the largest ratio of the number of M_{opt}-edges to that of M-edges is a length-five alternating path with three M_{opt}-edges and two M-edges, which has the ratio of 1.5. This proves the lemma.

Consider a Type-I component $(s_i, p_{j,t})$. Let ℓ_k be the lecturer who offers p_j. Since $p_{j,t}$ is not matched in M, p_j is under-subscribed in M. Then ℓ_k must be full in M since otherwise (s_i, p_j) blocks M. Because $p_{j,t}$ is matched in M_{opt} but not in M, we can find a vertex $p_{a,x}$ in V which is matched in M but not in M_{opt}, where p_a is offered by ℓ_k. We can remove $(s_i, p_{j,t})$ and add $(s_i, p_{a,x})$ to remove this Type-I component.

Consider a Type-II component $s_i - p_{a,x} - s_j - p_{b,y}$. Note that $p_a \neq p_b$ due to the construction of $G_{M,M_{opt}}$. Since s_i is unassigned in M, s_i is promoted. Then s_i applied to p_a when promoted, but was rejected. Therefore s_j must be promoted by Lemma 3.1(ii). This means that s_j applied to p_b at least once, but was rejected. Let ℓ_k be the lecturer who offers p_b. As mentioned several times before, this rejection can happen only when (1) p_b is full or (2) ℓ_k is full and p_b is ℓ_k's worst non-empty project or worse than that, and either (1) or (2) also holds for the output matching M. However $p_{b,y}$ is unmatched in M, so (2) must hold for M and hence ℓ_k is full in M. Since ℓ_k is full in M but $p_{b,y}$ is matched only in M_{opt}, there must be a vertex $p_{c,z}$ in V which is matched in M with some vertex, say s_d, but not matched in M_{opt}, where p_c is offered by ℓ_k. If we remove the edge $(s_j, p_{b,y})$ and add $(s_j, p_{c,z})$, then we will have an alternating path $s_i - p_{a,x} - s_j - p_{c,z} - s_d \cdots$ of length at least four. Hence this Type-II component is removed.

Note that in both of these cases, we used the property that ℓ_k is full in M. This implies that for each Type-I or Type-II component, we can find a distinct vertex in V which is matched only in M to perform the above mentioned replacement. We do this replacement for all Type-I and Type-II components in $G_{M,M_{opt}}$. This operation does not change any M-edges, so the number of students assigned to each lecturer or project in M is unchanged. In particular, a lecturer or a project full in M is still full in the modified graph.

As a result of these operations, we may still have a Type-II component. This can happen only when we removed a Type-I component, such as $(s_i, p_{j,t})$, using a length-two path, such as $p_{a,x} - s_r - p_{b,y}$, where $(s_r, p_{a,x})$ is an M-edge and $(s_r, p_{b,y})$ is an M_{opt}-edge. In this example, we removed $(s_i, p_{j,t})$ and added $(s_i, p_{a,x})$, and as a result we now have a Type-II component $s_i - p_{a,x} - s_r - p_{b,y}$. Note that p_a and p_j must be offered by the same lecturer, such as ℓ_k, because of the definition of the operation for Type-I components. Also, by the construction of $G_{M,M_{opt}}$, p_a and p_j must be different projects because $p_{j,t}$ is matched only in M_{opt} and $p_{a,x}$ is matched only in M.

If p_b is also offered by ℓ_k, then corresponding to the M_{opt}-edge $(s_r, p_{b,y})$, we can find a vertex $p_{c,z}$ in V which is matched in M but not in M_{opt}, where p_c is offered by ℓ_k, since ℓ_k is full in M. Then we can remove this Type-II component by replacing $(s_r, p_{b,y})$ with $(s_r, p_{c,z})$. Otherwise, let $\ell_{k'} (\neq \ell_k)$ be the lecturer who offers p_b. Suppose that s_r prefers p_b to p_a. Since p_b is under-subscribed in M, $\ell_{k'}$ must be full in M, since otherwise (s_r, p_b) blocks M. Then we can use the same argument as before to show the existence of a vertex $p_{c,z}$ which is matched in M but not in M_{opt}, where p_c is offered by $\ell_{k'}$, and we can replace $(s_r, p_{b,y})$ with $(s_r, p_{c,z})$. Suppose that s_r prefers p_a to p_b. If ℓ_k prefers p_a to p_j, then (s_r, p_a) blocks M_{opt}, a contradiction (note that $p_{a,x}$ is not matched in M_{opt} and hence p_a is under-subscribed in M_{opt}). If ℓ_k prefers p_j to p_a, then (s_i, p_j) blocks M, a contradiction. We have exhausted all of the cases, and have shown that all Type-I and Type-II components can be removed. This completes the proof. □

The following theorem follows immediately from Lemmas 3.5 and 3.6.

Theorem 3.7. SPA-P-APPROX-PROMOTION *is a 1.5-approximation algorithm for MAX-SPA-P.*

3.4. Tightness of the Analysis

We give an instance to show that our analysis of the approximation ratio is tight. There are three students s_1, s_2, and s_3 and one lecturer ℓ_1 with $d_1 = 3$.

Lecturer ℓ_1 offers three projects p_1, p_2, and p_3, where $c_1 = c_2 = c_3 = 1$. The preferences of the students and the lecturer are as follows:

$$\begin{array}{lll} s_1: & p_1 & \\ s_2: & p_1 & p_2 \\ s_3: & p_2 & p_3 \end{array} \qquad\qquad \begin{array}{llll} \ell_1: & p_3 & p_2 & p_1 \end{array}$$

Note that the matching $\{(s_1, p_1), (s_2, p_2), (s_3, p_3)\}$ of size three is stable, but the following execution of SPA-P-APPROX-PROMOTION yields a stable matching of size two $\{(s_2, p_1), (s_3, p_2)\}$:

1. s_1 applies to p_1 and is accepted.
2. s_3 applies to p_2 and is accepted.
3. s_2 applies to p_1 and is rejected.
4. s_2 applies to p_2 and is rejected.
5. s_2 is promoted.
6. s_2 applies to p_1 and is accepted; s_1 is rejected.
7. s_1 is promoted.
8. s_1 applies to p_1 and is rejected.

3.5. Promoting Many Times

In the course of SPA-P-APPROX-PROMOTION, each student is promoted at most once. One of the natural extensions is then to let a student be promoted more than once, where a student with more promotions is more preferred (within the same project). Unfortunately, however, we have a simple example to show that this extension does not improve the approximation ratio. There are three students s_1, s_2, s_3, three lecturers ℓ_1, ℓ_2, ℓ_3, and four projects, p_1 and p_2 offered by ℓ_1, p_3 offered by ℓ_2, and p_4 offered by ℓ_3. All of the capacities of lecturers and projects are one. Preference lists are defined as follows:

$$\begin{array}{lll} s_1: & p_2 & \\ s_2: & p_3 & p_1 \\ s_3: & p_3 & p_4 \end{array} \qquad\qquad \begin{array}{lll} \ell_1: & p_1 & p_2 \\ \ell_2: & p_3 & \\ \ell_3: & p_4 & \end{array}$$

First note that $\{(s_1, p_2), (s_2, p_3), (s_3, p_4)\}$ of size three is a maximum stable matching. Now, consider the following execution of the extended algorithm:

1. s_3 applies to p_3 and is accepted.
2. s_2 applies to p_3 and is rejected.
3. s_2 applies to p_1 and is accepted.

Then, afterwards, no matter how many times s_1 is promoted, s_1 will be rejected by p_2. Hence the extended algorithm produces a stable matching of size two. Note that this is also another tight example for Section 3.4.

4. Inapproximability

We first define the following optimization variant of the stable marriage problem, which we call *MAX-SMTI-1T* (abbreviation of "Maximum stable marriage problem with ties and incomplete lists with one-sided ties"). In an input, we have sets of men and women. Each man has an acceptable set of women, whom he is willing to be matched with, and has a preference list that orders his acceptable women in a strict order. Similarly, each woman has an acceptable set of men, and has a preference list for them. The women's preference lists may contain ties, meaning that two or more men in the same tie are considered to be of equal preference for her. A matching is a set of disjoint (man, woman)-pairs (m, w) such that m and w are acceptable to each other. If (m, w) is a pair in a matching M, we write $M(m) = w$ and $M(w) = m$. For a matching M, a (man, woman)-pair $(m, w) \notin M$ is a *blocking pair* if (i) m and w are acceptable to each other, (ii) either m is unmatched or prefers w to $M(m)$, and (iii) either w is unmatched or prefers m to $M(w)$. A matching that has no blocking pair is *stable*. MAX-SMTI-1T is the problem of finding a stable matching of maximum size. For an instance I of MAX-SMTI-1T, let $OPT(I)$ be the size of a maximum stable matching for I. The following proposition is obtained by letting $p = 1/3$ in Theorem 3.2 of [4].

Proposition 4.1. *[4] For any $\epsilon > 0$, if there is a polynomial-time algorithm that, given a MAX-SMTI-1T instance I with N men and N women, distinguishes between the following two cases, then P=NP.*

(1) $OPT(I) \geq \frac{7/3-\epsilon}{3}N$.
(2) $OPT(I) < \frac{19/9+\epsilon}{3}N$.

We prove a similar hardness for MAX-SPA-P by a reduction from MAX-SMTI-1T. For an instance I' of MAX-SPA-P, let $OPT(I')$ be the size of a maximum stable matching for I'.

Theorem 4.2. *For any $\epsilon > 0$, if there is a polynomial-time algorithm that, given a MAX-SPA-P instance I' with N' students, distinguishes between the following two cases, then P=NP.*

(1) $OPT(I') \geq \frac{7/3-\epsilon}{3}N'$.
(2) $OPT(I') < \frac{19/9+\epsilon}{3}N'$.

PROOF. Let I be a MAX-SMTI-1T instance with N men and N women. Without loss of generality, we assume that acceptability is symmetric, i.e., a man m includes a woman w in his list if and only if w includes m in her list. We construct a MAX-SPA-P instance I' with N' students. Our reduction satisfies conditions (i) $N' = N$ and (ii) $OPT(I') = OPT(I)$. Then it is not hard to see that Proposition 4.1 implies Theorem 4.2.

For each man m_i of I, we create a student s_i of I', and for each woman w_j of I, we create a lecturer ℓ_j of I'. For each woman w_j, let $T_{j,1}, T_{j,2}, \ldots, T_{j,t}$ be the ties in w_j's preference list in the order of preference, where a man not in a tie is considered as a tie of size one. Then, we create projects $p_{j,1}, p_{j,2}, \ldots, p_{j,t}$ that are offered by ℓ_j, where ℓ_j's preference list includes these projects in this order. Suppose that in I, a man m_i includes a woman w_j at the dth position in his list, and m_i is in a tie $T_{j,k}$ of woman w_j's list. Then, in I', student s_i includes the project $p_{j,k}$ at the dth position of the list. The capacity of each lecturer and each project is one. This completes the reduction. It is not hard to see that the reduction can be done in polynomial time. To illustrate the reduction, we give an example of MAX-SMTI-1T instance I in Fig. 1 and corresponding MAX-SPA-P instance I' in Fig. 2. In a woman's list in Fig. 1, men in the same tie are included in parenthesis.

m_1: w_1 w_3 w_2
m_2: w_3 w_1 w_2
m_3: w_4 w_1
m_4: w_3 w_2 w_1 w_4

w_1: $(m_1\ m_2)$ $(m_3\ m_4)$
w_2: m_2 m_1 m_4
w_3: m_1 $(m_2\ m_4)$
w_4: m_3 m_4

Figure 1: A MAX-SMTI-1T instance I

s_1: $p_{1,1}$ $p_{3,1}$ $p_{2,2}$
s_2: $p_{3,2}$ $p_{1,1}$ $p_{2,1}$
s_3: $p_{4,1}$ $p_{1,2}$
s_4: $p_{3,2}$ $p_{2,3}$ $p_{1,2}$ $p_{4,2}$

ℓ_1: $p_{1,1}$ $p_{1,2}$
ℓ_2: $p_{2,1}$ $p_{2,2}$ $p_{2,3}$
ℓ_3: $p_{3,1}$ $p_{3,2}$
ℓ_4: $p_{4,1}$ $p_{4,2}$

Figure 2: The MAX-SPA-P instance I' corresponding to I

Clearly condition (i) holds. In the rest of the proof, we show that condition (ii) holds. To see this, we show that (A) if there is a stable matching M of I, then there is a stable matching M' of I' such that $|M'| = |M|$, and (B) if there is a stable matching M' of I', then there is a stable matching M of I such that $|M| = |M'|$. The statement (A) implies $OPT(I') \geq OPT(I)$ and (B) implies $OPT(I) \geq OPT(I')$, which together implies condition (ii).

We show (A) first. Given a stable matching M of I, we create a matching M' of I' as follows: Suppose that a man m_i is matched with a woman w_j in M. Then, by construction, s_i's list includes a project $p_{j,k}$ offered by the lecturer ℓ_j, and such k is unique. In M', we assign a student s_i to $p_{j,k}$. If m_i is unmatched in M, then s_i is unassigned in M'. It is straightforward to check that M' satisfies all of the capacity constraints and that $|M'| = |M|$. Suppose that M' admits a blocking pair $(s_i, p_{j,k})$. Then, s_i is unassigned or prefers $p_{j,k}$ to $M'(s_i)$. Also, ℓ_j is unassigned, or assigned a student $s_{i'}$ to a project $p_{j,k'}$ and ℓ_j prefers $p_{j,k}$ to $p_{j,k'}$. Then, in M, m_i is unmatched or prefers w_j to $M(m_i)$, and w_j is unmatched or prefers m_i to $M(w_j)(= m_{i'})$, i.e., (m_i, w_j) is a blocking pair for M, contradicting the stability of M. Hence, we can conclude that M' admits no blocking pair.

Now suppose that M' admits a coalition $\{s_{i_0}, s_{i_1}, \ldots, s_{i_{r-1}}\}$, that is, s_{i_j} prefers $M'(s_{i_{j+1}})$ to $M'(s_{i_j})$ for each j. Then, remove this coalition from M', that is, reassign s_{i_j} to $M'(s_{i_{j+1}})$ for each j. Note that an application of this operation does not change the matching size. Also, note that no new blocking pair is created because no one becomes worse off. We apply this operation as long as there is a coalition. This sequence of operations must terminate in finite number of steps because at least two students become better off by one application. Hence at the termination, we have a stable matching of size $|M'|$.

Next we show (B). Let M' be a stable matching for I'. We construct a matching M of I as follows: If a student s_i is assigned to a project $p_{j,k}$ in M', then a man m_i is matched with a woman w_j in M (note that w_j is in m_i's list by construction). If s_i is unassigned in M', then m_i is unmatched in M. Again, it is easy to see that M is a stable matching and $|M| = |M'|$. This completes the proof. $\square$

Corollary 4.3. *Assume that P≠NP. Then for any constant $\delta > 0$, there is no polynomial-time $(21/19 - \delta)$-approximation algorithm for MAX-SPA-P.*

Proof. By Theorem 4.2, we see that the existence of a polynomial-time algorithm that distinguishes between the following two cases implies P=NP for an arbitrary small positive constant ϵ:

(1) $OPT(I') \geq \frac{7/3-\epsilon}{3}N'$.
(2) $OPT(I') < \frac{19/9+\epsilon}{3}N'$.

Suppose that there is a polynomial-time approximation algorithm T for MAX-SPA-P whose approximation ratio is at most $21/19 - \delta$ for some δ. Then consider these conditions with fixed constant ϵ such that $\epsilon < \frac{361\delta}{360-171\delta}$.

If an instance of Case (1) is given to T, it outputs a solution whose size is at least $\frac{7/3-\epsilon}{3}N'\frac{1}{21/19-\delta}$. If an instance of Case (2) is given to T, it outputs a solution whose size is less than $\frac{19/9+\epsilon}{3}N'$. Since $\frac{7/3-\epsilon}{3}N'\frac{1}{21/19-\delta} > \frac{19/9+\epsilon}{3}N'$ from the definition of ϵ, T can distinguish between Cases (1) and (2), which implies P=NP. This completes the proof. □

As mentioned in Remark 3.6 of [4], MAX-SMTI-1T is hard to approximate within $1.25-\delta$ for any positive constant δ if Minimum Vertex Cover problem is hard to approximate within $2-\epsilon$ for any positive constant ϵ (note that the "if-part" is true if the Unique Games Conjecture is true [6]). Using the reduction in the proof of Theorem 4.2, we can prove the same hardness for MAX-SPA-P:

Theorem 4.4. *Assume that, for any positive constant ϵ, there is no polynomial-time $(2-\epsilon)$-approximation algorithm for Minimum Vertex Cover problem. Then, for any positive constant δ, there is no polynomial-time $(1.25-\delta)$-approximation algorithm for MAX-SPA-P.*

Proof. Suppose that, for some δ', there is a polynomial-time $(1.25-\delta')$-approximation algorithm A for MAX-SPA-P. Then, the following algorithm B is a polynomial-time $(1.25-\delta')$-approximation algorithm for MAX-SMTI-1T: Given an instance I of MAX-SMTI-1T, B first translates it to an instance I' of MAX-SPA-P using the reduction in the proof of Theorem 4.2. It then solves I' using A and obtains a solution M', and transforms it to a solution M of I in the same manner as given in the proof of Theorem 4.2. Then, by Remark 3.6 of [4], there is a polynomial-time $(2-\epsilon')$-approximation algorithm for Minimum Vertex Cover problem for some ϵ', which contradicts our assumption. □

5. Conclusions

In this paper, we improved the upper and lower bounds on the approximation ratio for MAX-SPA-P. One research direction is to further improve the upper bound. For example, a state of the art approximation algorithm for MAX-SMTI-1T [5] generalizes Király's idea [7] using a Linear Programming approach. Its approximation ratio of $25/17(\simeq 1.4706)$ is slightly better than 1.5. One possible next step is to verify whether this idea can be applied to SPA-P-APPROX-PROMOTION.

Acknowledgments. The authors would like to thank the anonymous reviewer for the valuable comments.

References

[1] D. J. Abraham, R. W. Irving and D. F. Manlove, Two algorithms for the Student-Project Allocation problem, Journal of Discrete Algorithms, 5 (1) (2007) 73–90.

[2] D. Gale and L. S. Shapley, College admissions and the stability of marriage, Amer. Math. Monthly 69 (1962) 9–15.

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

[4] M. M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa, "Improved approximation results for the stable marriage problem," ACM Transactions on Algorithms 3 (3) (2007) Article No. 30.

[5] K. Iwama, S. Miyazaki, and H. Yanagisawa, A 25/17-approximation algorithm for the stable marriage problem with one-sided ties, in: Proceedings of ESA 2010: the 18th Annual Europian Symposium on Algorithms, in: Lecture Notes in Computer Science, vol. 6347, Springer-Verlag (2010) 135–146.

[6] S. Khot and O. Regev, Vertex cover might be hard to approximate to within $2 - \epsilon$, Journal of Computer and System Sciences 74 (3) (2008) 335–349.

[7] Z. Király, Better and simpler approximation algorithms for the stable marriage problem, Algorithmica 60 (1) (2011) 3–20.

[8] D. F. Manlove, and G. O'Malley, Student-project allocation with preferences over projects, Journal of Discrete Algorithms 6 (4) (2008) 553–560.

Algorithm 1 SPA-P-APPROX-PROMOTION

```
M := ∅.
Let all students be unpromoted.
while (there exists an unassigned student s_i such that s_i's list is non-empty or s_i is unpromoted) do
    if (s_i's list is empty and s_i is unpromoted) then
        Promote s_i.
    end if
    p_j := first project on s_i's list.
    ℓ_k := lecturer who offers p_j.
    /* s_i applies to p_j */
    if (A. (p_j is full) or (ℓ_k is full and p_j is ℓ_k's worst non-empty project)) then
        if ((s_i is unpromoted) or (there is no unpromoted student in M(p_j))) then
            Reject s_i.
        else
            Reject an arbitrary unpromoted student in M(p_j) and add (s_i, p_j) to M.
        end if
    else if (B. ℓ_k is full and prefers ℓ_k's worst non-empty project to p_j) then
        Reject s_i.
    else if (C. Otherwise) then
        Add (s_i, p_j) to M.
        if (ℓ_k is over-subscribed) then
            p_z := ℓ_k's worst non-empty project. /* note that ℓ_k prefers p_j to p_z. */
            if (M(p_z) contains an unpromoted student) then
                Reject an arbitrary unpromoted student in M(p_z).
            else
                Reject an arbitrary student in M(p_z).
            end if
        end if
    end if
end while
Return M.
```