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

Kazuo Iwama[1], Shuichi Miyazaki[2], Hiroki Yanagisawa[3]

[1] Graduate School of Informatics, Kyoto University
`iwama@kuis.kyoto-u.ac.jp`
[2] Academic Center for Computing and Media Studies, Kyoto University
`shuichi@media.kyoto-u.ac.jp`
[3] IBM Research - Tokyo
`yanagis@jp.ibm.com`

Abstract. Manlove and O'Malley [9] 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).

1 Introduction

Assignment problems based on the preferences of participants, which originated from the famous *Hospitals/Residents problem* (*HR*) [4], are important 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 [3].

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* (called a quota in the original HR). 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 [9] 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) (assuming P $\neq$ NP). For the upper bound, we modify Manlove and O'Malley's algorithm SPA-P-APPROX [9] 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 the Minimum Vertex Cover problem, which is similar to the one used in [5] to prove the approximation lower bound for MAX-SMTI.

2 Preliminaries

Here we give a formal definition of SPA-P and MAX-SPA-P, derived directly from the literature [9]. 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*, 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 $r \in S \cup P \cup L$, let $M(r)$ be the set of assignees of 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 which s_i is assigned to. 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 the 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 [9] 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 $\max\{opt(x)/A(x)\} \leq r$ over all instances x, where $opt(x)$ and $A(x)$ are the sizes of the optimal and the algorithm's solutions, respectively.

3 Approximability

3.1 Algorithm SPA-P-APPROX-PROMOTION

Manlove and O'Malley's algorithm SPA-P-APPROX [9] 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.

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:

Algorithm 1 SPA-P-APPROX-PROMOTION

$M := \emptyset$.
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 s_i'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 $p_z \neq p_j$.)
      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.

Lemma 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 new 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 from p_a, p_a must be ℓ_k's worst non-empty project before the rejection. Then after this rejection, p_a is still ℓ_k's worst non-empty project or p_a

becomes worse than it was (if s was the only student assigned to p_a). 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 from 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 it was. Then we can apply the same argument as in Case (2). □

The proof of the following lemma is basically similar and is omitted.

Lemma 2. *Suppose that, during the execution of* SPA-P-APPROX-PROMOTION, *a project p_a has rejected a student. Then after that point, no new unpromoted student can be accepted for p_a.*

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

Lemma 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 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 2. Then s_{i_1} was rejected from p_{i_2} when s_{i_1} was promoted. This means that s_{i_2} is eventually promoted by Lemma 1(ii). Repeating this argument, we can conclude that $s_{i_{r-1}}$ is eventually promoted. Then this contradicts Lemma 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, p_{i_1} accepts no new students by Lemma 1(i). This contradicts the fact that s_{i_1} was accepted to p_{i_1} later. □

Lemma 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 from 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. □

The following lemma follows immediately from Lemmas 3 and 4.

Lemma 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 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 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. Therefore, 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 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 unassigned in M, so only (2) is possible. Since ℓ_k is full in M, there must be 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. We can remove the edge $(s_j, p_{b,y})$ and add $(s_j, p_{c,z})$ to remove this Type-II component.

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 $(s_i, p_{a,x})$. 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 $\ell_{k'}$. 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 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 5 and 6.

Theorem 1. 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 $d_1 = 3$ who offers three projects p_1, p_2, and p_3 with $c_1 = c_2 = c_3 = 1$. The preferences of the students and the lecturer are as follows:

$$\begin{array}{ll} s_1\colon p_1 & \qquad \ell_1\colon p_3 \;\; p_2 \;\; p_1 \\ s_2\colon p_1 \;\; p_2 & \\ s_3\colon p_2 \;\; p_3 & \end{array}$$

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.

4 Inapproximability

The stable marriage problem (SM) [3, 4] is the problem of finding a stable matching, given sets of men and women and each person's preference list over the members of the opposite gender. If ties are allowed in the preference lists and if the preference lists may be incomplete (i.e., unacceptable persons may be dropped from the lists), then the problem of finding a maximum stable matching (MAX-SMTI) is NP-hard even if ties appear on only one side (e.g., the men's lists must be totally ordered) [8]. We call this restricted problem *MAX-SMTI-1T*.

There is a similarity between MAX-SMTI-1T and MAX-SPA-P, so we can define the following natural reduction from MAX-SMTI-1T to MAX-SPA-P: Suppose that in the MAX-SMTI-1T instance I, the men's lists are strict and the women's lists may contain ties. Then in the MAX-SPA-P instance I', the students and lecturers correspond to men and women in I, respectively. For each woman w's list, we create a project for each tie in the list, where a man not in a tie is considered as a tie of size one. These projects are offered by the lecturer ℓ_w corresponding to the woman w, and the order of projects in ℓ_w's list is consistent with w's list in I. Each project p is acceptable to the students corresponding to the men in the tie associated with this project p. The order of projects in the preference list of a student is naturally generated from corresponding man's (strictly ordered) list in I. The capacity of each lecturer and each project is one.

Using the above reduction, we can prove that the sizes of a maximum stable matching of I and a maximum blocking-pair-free matching of I' coincide. The only problem is that there is a coalition-freeness condition in the stability definition of SPA-P. Therefore a reduction from the general instances of MAX-SMTI-1T to MAX-SPA-P cannot be applied. However, it turns out that if we use only the instances generated by the reduction in [5], then this problem can be resolved and the sizes of the optimal solutions for MAX-SMTI-1T and MAX-SPA-P coincide, so that the approximation lower bound of 21/19 for MAX-SMTI-1T proved in [5] applies to MAX-SPA-P. For the completeness of this article, however, we give a direct reduction from the *Minimum Vertex Cover* problem (*MVC*) to MAX-SPA-P.

For a graph $G = (V, E)$, a subset $C \subseteq V$ of vertices is called a *vertex cover* for G if for any edge, at least one of its endpoints is in C. MVC is the problem of finding a vertex cover of minimum size for a given graph. Let $OPT(G)$ be the size of a minimum vertex cover for G. We can now use the well-known Proposition 1.

Proposition 1. *[2] For any $\epsilon > 0$ and $p < \frac{3-\sqrt{5}}{2}$, if there is a polynomial-time algorithm that, given a graph $G = (V, E)$, distinguishes between these two cases, then P=NP.*

(1) $OPT(G) \leq (1 - p + \epsilon)|V|$.
(2) $OPT(G) > (1 - \max\{p^2, 4p^3 - 3p^4\} - \epsilon)|V|$.

For an instance I of MAX-SPA-P, let $OPT(I)$ be the size of a maximum stable matching for I. Then we can prove Theorem 2.

Theorem 2. *For any $\epsilon > 0$ and $p < \frac{3-\sqrt{5}}{2}$, if there is a polynomial-time algorithm that, given a MAX-SPA-P instance I of N students, distinguishes between these two cases, then P=NP.*

(1) $OPT(I) \geq \frac{2+p-\epsilon}{3}N$.
(2) $OPT(I) < \frac{2+\max\{p^2,4p^3-3p^4\}+\epsilon}{3}N$.

Proof. Given a graph $G = (V, E)$, we will construct, in polynomial time, an instance I_G of MAX-SPA-P with N students. Our reduction satisfies conditions (i) $N = 3|V|$ and (ii) $OPT(I_G) = 3|V| - OPT(G)$. Then it is not hard to see that Proposition 1 implies Theorem 2.

Now we show the reduction. For each vertex v_i of G, we construct three students a_i, b_i, and c_i and three lecturers x_i, y_i, and z_i. Suppose that v_i is adjacent to k vertices $v_{i_1}, v_{i_2}, \cdots, v_{i_k}$ $(i_1 < i_2 < \cdots < i_k)$. Then we construct $k + 4$ projects X_i, Y_i, $Z_{i,-}$, Z_{i,i_1}, $\cdots$, Z_{i,i_k} and $Z_{i,+}$, where X_i is offered by x_i, Y_i by y_i, and $Z_{i,-}$, Z_{i,i_1}, $\cdots$, Z_{i,i_k}, $Z_{i,+}$ by z_i. The capacity of each project and each lecturer is one.

Next, we define the acceptability of projects to students. The project X_i is acceptable to only one student a_i. The project Y_i is acceptable to two students

a_i and b_i. $Z_{i,-}$ is acceptable to only b_i, and $Z_{i,+}$ is acceptable to only c_i. For each $j = 1, 2, \ldots, k$, the project Z_{i,i_j} is acceptable to only one student a_{i_j} (corresponding to the adjacent vertex v_{i_j}). Finally, we define preference lists of the students and lecturers corresponding to v_i as:

$$\begin{array}{ll} a_i\colon Y_i\ Z_{i_1,i}\ Z_{i_2,i}\ \cdots\ Z_{i_k,i}\ X_i & \quad x_i\colon X_i \\ b_i\colon Y_i\ Z_{i,-} & \quad y_i\colon Y_i \\ c_i\colon Z_{i,+} & \quad z_i\colon Z_{i,-}\ Z_{i,i_1}\ \cdots\ Z_{i,i_k}\ Z_{i,+} \end{array}$$

Obviously, this reduction can be performed in polynomial time. Since the capacities of all of the projects and lecturers are one, for a project or a lecturer r assigned in M, we may use $M(r)$ to denote the unique student assigned to r. 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 vertex cover C of G, then there is a stable matching M of I_G such that $|M| = 3|V| - |C|$, and (B) if there is a stable matching M of I_G, then there is a vertex cover C of G such that $|C| = 3|V| - |M|$. The statement (A) implies $OPT(I_G) \geq 3|V| - OPT(G)$ and (B) implies $OPT(G) \leq 3|V| - OPT(I_G)$, which together implies condition (ii).

We show (A) first. Given a vertex cover C for G, we construct a stable matching M for I_G as follows: For each vertex v_i, if $v_i \in C$, let $M(a_i) = Y_i$, $M(b_i) = Z_{i,-}$, and leave c_i unassigned. If $v_i \notin C$, let $M(a_i) = X_i$, $M(b_i) = Y_i$, and $M(c_i) = Z_{i,+}$. Since the capacity of each lecturer is one, we can regard M as a matching between students and lecturers. Fig. 1 shows a part of M corresponding to v_i. By an easy calculation, we can see that $|M| = 2|C| + 3(|V| - |C|) = 3|V| - |C|$ as required.

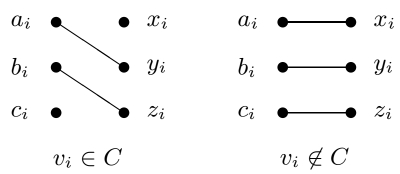

Fig. 1. A part of matching M

We will show that M is stable. We first show that there is no blocking pair. For $v_i \in C$, a_i is assigned to the top project, so that a_i cannot be part of a blocking pair. The student b_i is assigned to the second project, but the first project Y_i and the lecturer y_i who offers Y_i are both full and hence b_i cannot form a blocking pair. Student c_i is unassigned but the lecturer z_i, who offers c_i's only acceptable project $Z_{i,+}$, is full and prefers c_i's assigned project $Z_{i,-}$ to $Z_{i,+}$, so that c_i cannot be part of a blocking pair. For $v_i \notin C$, b_i and c_i are assigned to the top projects respectively. The only possibility is that a_i forms a blocking pair with some project among Y_i, $Z_{i_1,i}$, $Z_{i_2,i}$, $\cdots$, $Z_{i_k,i}$, but it is easy to see that Y_i is excluded. Therefore, suppose that a_i forms a blocking pair with $Z_{i_j,i}$ for some j. Then by construction there is an edge between v_i and v_{i_j}, and the lecturer z_{i_j} is assigned the student c_{i_j} for the project $Z_{i_j,+}$ (since in the other case, z_{i_j} receives a student for the most preferred project and hence $(a_i, Z_{i_j,i})$ cannot be a blocking pair). This means that $v_{i_j} \notin C$ by the construction of M. Then this contradicts the assumption that C is a vertex cover for G. We then show that M admits no coalition. Note that in M, each student corresponding to the vertex v_i of G is assigned to a project corresponding to v_i. This implies that any coalition must consist of students and projects corresponding to the same vertex. However we can easily verify that there is no coalition in either the case of $v_i \in C$ or $v_i \notin C$, which completes the stability proof.

Next we show (B). Let M be a stable matching for I_G. First, if the project Y_i is unassigned, then both (a_i, Y_i) and (b_i, Y_i) block M, which is a contradiction. Therefore either $M(Y_i) = a_i$ or $M(Y_i) = b_i$.

First, suppose that $M(Y_i) = a_i$. Then $M(b_i) = Z_{i,-}$ since otherwise, $(b_i, Z_{i,-})$ blocks M. Then c_i is unassigned and x_i and X_i are empty in M. In this case, we say that v_i causes *Pattern 1.* A diagrammatic representation of Pattern 1 is given in Fig. 2.

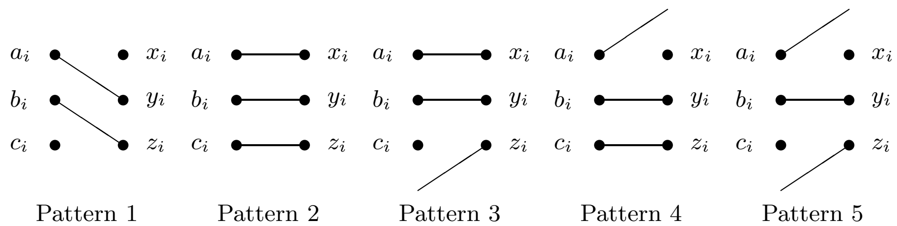

Fig. 2. Five patterns caused by v_i

Next, suppose that $M(Y_i) = b_i$. Then a_i is assigned in M, since otherwise (a_i, X_i) blocks M. Since Y_i is already taken by b_i, there remain two cases: (a) $M(a_i) = X_i$ and (b) $M(a_i) = Z_{i_j,i}$ for some j. Similarly, if z_i is empty in M, then $(c_i, Z_{i,+})$ blocks M. This means either (c) $M(z_i) = c_i$ or (d) $M(z_i) = a_{i_j}$ for some j. Hence, we have a total of four cases. These cases are referred to as Patterns 2 through 5 (see Fig. 2). For example, a combination of cases (b) and (c) corresponds to Pattern 4. Lemma 7, whose proof is omitted by the space restriction, excludes the possibility of Patterns 3 or 4.

Lemma 7. *Each vertex causes Pattern 1, 2 or 5.*

By Lemma 7, each vertex v_i will lead to Pattern 1, 2, or 5. We construct the subset C of vertices in this way: If v_i causes Pattern 1 or 5, then let $v_i \in C$, otherwise, let $v_i \notin C$.

We show that C is a vertex cover for G. Assume that C is not a vertex cover for G. Then there are two vertices v_i and v_j in $V \setminus C$ such that $(v_i, v_j) \in E$

and both of them cause Pattern 2. Then both $(a_i, Z_{j,i})$ and $(a_j, Z_{i,j})$ block M, contradicting the stability of M. Hence, C is a vertex cover for G. It is obvious that $|M| = 2|C| + 3(|V| - |C|) = 3|V| - |C|$. Hence, statement (B) holds. This completes the proof of Theorem 2. □

By letting $p = \frac{1}{3}$ in Theorem 2, we have Corollary 1.

Corollary 1. *Assume that P≠NP. Then for any constant* $\delta > 0$*, there is no polynomial-time* $(21/19 - \delta)$*-approximation algorithm for MAX-SPA-P.*

Remark. Using the same argument as Remark 3.6 of [5], we can claim that MAX-SPA-P is hard to approximate within $1.25 - \delta$ if MVC is hard to approximate within $2 - \epsilon$ (where δ and ϵ are arbitrary positive constants).

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 recent approximation algorithm for MAX-SMTI-1T [6] generalizes Király's idea [7] using 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 anonymous reviewers for their valuable comments. This work was supported by KAKENHI 22240001 and 20700009.

References

1. D. J. Abraham, R. W. Irving and D. F. Manlove, "Two algorithms for the Student-Project Allocation problem," *J. Discrete Algorithms*, Vol. 5, No. 1, pp. 73–90, 2007.
2. I. Dinur and S. Safra, "On the hardness of approximating minimum vertex-cover," *Annals of Mathematics*, Vol. 162(1), pp. 439–485, 2005.
3. D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, Vol. 69, pp. 9–15, 1962.
4. D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Boston, MA, 1989.
5. M. M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa, "Improved approximation results for the stable marriage problem," *ACM Transactions on Algorithms*, Vol. 3(3), Article No. 30, 2007.
6. K. Iwama, S. Miyazaki, and H. Yanagisawa, "A 25/17-approximation algorithm for the stable marriage problem with one-sided ties," *Proc. ESA 2010*, LNCS 6347, pp. 135–146, 2010.
7. Z. Király, "Better and simpler approximation algorithms for the stable marriage problem," *Algorithmica*, DOI 10.1007/s00453-009-9371-7, 2009.
8. D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. "Hard variants of stable marriage," *Theoretical Computer Science*, Vol. 276(1–2), pp. 261–279, 2002.
9. D. F. Manlove, and G. O'Malley, "Student-project allocation with preferences over projects," *Journal of Discrete Algorithms*, Vol. 6, pp. 553–560, 2008.