# An Integer Programming Approach to the Student-Project Allocation Problem with Preferences over Projects*

David Manlove[0000−0001−6754−7308], Duncan Milne, and Sofiat Olaosebikan[0000−0002−8003−7887]

School of Computing Science, University of Glasgow,
e-mail: David.Manlove@glasgow.ac.uk, Duncan.Milne1@gmail.com, s.olaosebikan.1@research.gla.ac.uk

**Abstract.** The Student-Project Allocation problem with preferences over Projects (SPA-P) involves sets of students, projects and lecturers, where the students and lecturers each have preferences over the projects. In this context, we typically seek a stable matching of students to projects (and lecturers). However, these stable matchings can have different sizes, and the problem of finding a maximum stable matching (MAX-SPA-P) is NP-hard. There are two known approximation algorithms for MAX-SPA-P, with performance guarantees of 2 and $\frac{3}{2}$. In this paper, we describe an Integer Programming (IP) model to enable MAX-SPA-P to be solved optimally. Following this, we present results arising from an empirical analysis that investigates how the solution produced by the approximation algorithms compares to the optimal solution obtained from the IP model, with respect to the size of the stable matchings constructed, on instances that are both randomly-generated and derived from real datasets. Our main finding is that the $\frac{3}{2}$-approximation algorithm finds stable matchings that are very close to having maximum cardinality.

## 1 Introduction

Matching problems, which generally involve the assignment of a set of agents to another set of agents based on preferences, have wide applications in many real-world settings. One such application can be seen in an educational context, e.g., the allocation of pupils to schools, school-leavers to universities and students to projects. In the context of allocating students to projects, university lecturers propose a range of projects, and each student is required to provide a preference over the available projects that she finds acceptable. Lecturers may also have preferences over the students that find their project acceptable and/or the projects that they offer. There may also be upper bounds on the number of

* The first author was supported by grant EP/P028306/1 from the Engineering and Physical Sciences Research Council, and the third author was supported by a College of Science and Engineering Scholarship, University of Glasgow.

students that can be assigned to a particular project, and the number of students that a given lecturer is willing to supervise. The problem then is to allocate students to projects based on these preferences and capacity constraints – the so-called *Student-Project Allocation problem* (SPA) [3,11].

Two major models of SPA exist in the literature: one permits preferences only from the students [2,6,10,14], while the other permits preferences from the students and lecturers [1,8]. Given the large number of students that are typically involved in such an allocation process, many university departments seek to automate the allocation of students to projects. Examples include the School of Computing Science, University of Glasgow [10], the Faculty of Science, University of Southern Denmark [4], the Department of Computing Science, University of York [8], and elsewhere [2,3,6,16].

In general, we seek a *matching*, which is a set of agent pairs who find one another acceptable that satisfies the capacities of the agents involved. For matching problems where preferences exist from the two sets of agents involved (e.g., junior doctors and hospitals in the classical *Hospitals-Residents problem* (HR) [5], or students and lecturers in the context of SPA), it has been argued that the desired property for a matching one should seek is that of *stability* [15]. Informally, a *stable matching* ensures that no acceptable pair of agents who are not matched together would rather be assigned to each other than remain with their current assignees.

Abraham, Irving and Manlove [1] proposed two linear-time algorithms to find a stable matching in a variant of SPA where students have preferences over projects, whilst lecturers have preferences over students. The stable matching produced by the first algorithm is student-optimal (that is, students have the best possible projects that they could obtain in any stable matching) while the one produced by the second algorithm is lecturer-optimal (that is, lecturers have the best possible students that they could obtain in any stable matching).

Manlove and O'Malley [12] proposed another variant of SPA where both students and lecturers have preferences over projects, referred to as SPA-P. In their paper, they formulated an appropriate stability definition for SPA-P, and they showed that stable matchings in this context can have different sizes. Moreover, in addition to stability, a very important requirement in practice is to match as many students to projects as possible. Consequently, Manlove and O'Malley [12] proved that the problem of finding a maximum cardinality stable matching, denoted MAX-SPA-P, is NP-hard. Further, they gave a polynomial-time 2-approximation algorithm for MAX-SPA-P. Subsequently, Iwama, Miyazaki and Yanagisawa [7] described an improved approximation algorithm with an upper bound of $\frac{3}{2}$, which builds on the one described in [12]. In addition, Iwama *et al.* [7] showed that MAX-SPA-P is not approximable within $\frac{21}{19} - \epsilon$, for any $\epsilon > 0$, unless P = NP. For the upper bound, they modified Manlove and O'Malley's algorithm [12] using Király's idea [9] for the approximation algorithm to find a maximum stable matching in a variant of the *Stable Marriage problem*.

Considering the fact that the existing algorithms for MAX-SPA-P are only guaranteed to produce an approximate solution, we seek another technique to

enable MAX-SPA-P to be solved optimally. Integer Programming (IP) is a powerful technique for producing optimal solutions to a range of NP-hard optimisation problems, with the aid of commercial optimisation solvers, e.g., Gurobi [17], GLPK [18] and CPLEX [19]. These solvers can allow IP models to be solved in a reasonable amount of time, even with respect to problem instances that occur in practical applications.

***Our Contribution.*** In Sect. 3, we describe an IP model to enable MAX-SPA-P to be solved optimally, and present a correctness result. In Sect. 4, we present results arising from an empirical analysis that investigates how the solution produced by the approximation algorithms compares to the optimal solution obtained from our IP model, with respect to the size of the stable matchings constructed, on instances that are both randomly-generated and derived from real datasets. These real datasets are based on actual student preference data and manufactured lecturer preference data from previous runs of student-project allocation processes at the School of Computing Science, University of Glasgow. We also present results showing the time taken by the IP model to solve the problem instances optimally. Our main finding is that the $\frac{3}{2}$-approximation algorithm finds stable matchings that are very close to having maximum cardinality. The next section gives a formal definition for SPA-P.

## 2 Definitions and Preliminaries

We give a formal definition for SPA-P as described in the literature [12]. An instance $\mathtt{I}$ of SPA-P involves a set $\mathcal{S} = \{s_1, s_2, \ldots, s_{n_1}\}$ of *students*, a set $\mathcal{P} = \{p_1, p_2, \ldots, p_{n_2}\}$ of *projects* and a set $\mathcal{L} = \{l_1, l_2, \ldots, l_{n_3}\}$ of *lecturers*. Each lecturer $l_k \in \mathcal{L}$ offers a non-empty subset of projects, denoted by $P_k$. We assume that $P_1, P_2, \ldots, P_{n_3}$ partitions $\mathcal{P}$ (that is, each project is offered by one lecturer). Also, each student $s_i \in \mathcal{S}$ has an *acceptable* set of projects $A_i \subseteq \mathcal{P}$. We call a pair $(s_i, p_j) \in \mathcal{S} \times \mathcal{P}$ an *acceptable pair* if $p_j \in A_i$. Moreover $s_i$ ranks $A_i$ in strict order of preference. Similarly, each lecturer $l_k$ ranks $P_k$ in strict order of preference. Finally, each project $p_j \in \mathcal{P}$ and lecturer $l_k \in \mathcal{L}$ has a positive capacity denoted by $c_j$ and $d_k$ respectively.

An *assignment* $M$ is a subset of $\mathcal{S} \times \mathcal{P}$ where $(s_i, p_j) \in M$ implies that $s_i$ finds $p_j$ acceptable (that is, $p_j \in A_i$). We define the *size* of $M$ as the number of (student, project) pairs in $M$, denoted $|M|$. If $(s_i, p_j) \in M$, we say that $s_i$ is *assigned to* $p_j$ and $p_j$ is *assigned* $s_i$. Furthermore, we denote the project assigned to student $s_i$ in $M$ as $M(s_i)$ (if $s_i$ is unassigned in $M$ then $M(s_i)$ is undefined). Similarly, we denote the set of students assigned to project $p_j$ in $M$ as $M(p_j)$. For ease of exposition, if $s_i$ is assigned to a project $p_j$ offered by lecturer $l_k$, we may also say that $s_i$ is *assigned to* $l_k$, and $l_k$ is *assigned* $s_i$. Thus we denote the set of students assigned to $l_k$ in $M$ as $M(l_k)$.

A project $p_j \in \mathcal{P}$ is *full*, *undersubscribed* or *oversubscribed* in $M$ if $|M(p_j)|$ is equal to, less than or greater than $c_j$, respectively. The corresponding terms apply to each lecturer $l_k$ with respect to $d_k$. We say that a project $p_j \in \mathcal{P}$ is *non-empty* if $|M(p_j)| > 0$.

A *matching* $M$ is an assignment such that $|M(s_i)| \leq 1$ for each $s_i \in \mathcal{S}$, $|M(p_j)| \leq c_j$ for each $p_j \in \mathcal{P}$, and $|M(l_k)| \leq d_k$ for each $l_k \in \mathcal{L}$ (that is, each student is assigned to at most one project, and no project or lecturer is oversubscribed). Given a matching $M$, an acceptable pair $(s_i, p_j) \in (\mathcal{S} \times \mathcal{P}) \setminus M$ is a *blocking pair* of $M$ if the following conditions are satisfied:

1. either $s_i$ is unassigned in $M$ or $s_i$ prefers $p_j$ to $M(s_i)$, and $p_j$ is undersubscribed, and either
   (a) $s_i \in M(l_k)$ and $l_k$ prefers $p_j$ to $M(s_i)$, or
   (b) $s_i \notin M(l_k)$ and $l_k$ is undersubscribed, or
   (c) $s_i \notin M(l_k)$ and $l_k$ prefers $p_j$ to his worst non-empty project,
   where $l_k$ is the lecturer who offers $p_j$.

If such a pair were to occur, it would undermine the integrity of the matching as the student and lecturer involved would rather be assigned together than remain in their current assignment. With respect to the SPA-P instance given in Fig. 1, $M_1 = \{(s_1, p_3), (s_2, p_1)\}$ is clearly a matching. It is obvious that each of students $s_1$ and $s_2$ is matched to her first ranked project in $M_1$. Although $s_3$ is unassigned in $M_1$, the lecturer offering $p_3$ (the only project that $s_3$ finds acceptable) is assumed to be indifferent among those students who find $p_3$ acceptable. Also $p_3$ is full in $M_1$. Thus, we say that $M_1$ admits no blocking pair.

| Student preferences | | | | Lecturer preferences | | |
|---|---|---|---|---|---|---|
| $s_1$: | $p_3$ | $p_2$ | $p_1$ | $l_1$: | $p_2$ | $p_1$ |
| $s_2$: | $p_1$ | $p_2$ | | $l_2$: | $p_3$ | |
| $s_3$: | $p_3$ | | | | | |

**Fig. 1.** An instance $\mathtt{I}_1$ of SPA-P. Each project has capacity 1, whilst each of lecturer $l_1$ and $l_2$ has capacity 2 and 1 respectively.

Another way in which a matching could be undermined is through a group of students acting together. Given a matching $M$, a *coalition* is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$ such that each student $s_{i_j}$ $(0 \leq j \leq r-1)$ is assigned in $M$ and prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$, where addition is performed modulo $r$. With respect to Fig. 1, the matching $M_2 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3)\}$ admits a coalition $\{s_1, s_2\}$, as students $s_1$ and $s_2$ would rather permute their assigned projects in $M_2$ so as to be better off. We note that the number of students assigned to each project and lecturer involved in any such swap remains the same after such a permutation. Moreover, the lecturers involved would have no incentive to prevent the switch from occurring since they are assumed to be indifferent between the students assigned to the projects they are offering. If a matching admits no coalition, we define such matching to be *coalition-free*.

Given an instance $\mathtt{I}$ of SPA-P, we define a matching $M$ in $\mathtt{I}$ to be *stable* if $M$ admits no blocking pair and is coalition-free. It turns out that with respect to this definition, stable matchings in $\mathtt{I}$ can have different sizes. Clearly, each

of the matchings $M_1 = \{(s_1, p_3), (s_2, p_1)\}$ and $M_3 = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$ is stable in the SPA-P instance $\mathtt{I}_1$ shown in Fig. 1. The varying sizes of the stable matchings produced naturally leads to the problem of finding a maximum cardinality stable matching given an instance of SPA-P, which we denote by MAX-SPA-P. In the next section, we describe our IP model to enable MAX-SPA-P to be solved optimally.

## 3 An IP model for MAX-SPA-P

Let $\mathtt{I}$ be an instance of SPA-P involving a set $\mathcal{S} = \{s_1, s_2, \ldots, s_{n_1}\}$ of students, a set $\mathcal{P} = \{p_1, p_2, \ldots, p_{n_2}\}$ of projects and a set $\mathcal{L} = \{l_1, l_2, \ldots, l_{n_3}\}$ of lecturers. We construct an IP model $\mathtt{J}$ of $\mathtt{I}$ as follows. Firstly, we create binary variables $x_{i,j} \in \{0, 1\}$ $(1 \leq i \leq n_1, 1 \leq j \leq n_2)$ for each acceptable pair $(s_i, p_j) \in \mathcal{S} \times \mathcal{P}$ such that $x_{i,j}$ indicates whether $s_i$ is assigned to $p_j$ in a solution or not. Henceforth, we denote by $S$ a solution in the IP model $\mathtt{J}$, and we denote by $M$ the matching derived from $S$. If $x_{i,j} = 1$ under $S$ then intuitively $s_i$ is assigned to $p_j$ in $M$, otherwise $s_i$ is not assigned to $p_j$ in $M$. In what follows, we give the constraints to ensure that the assignment obtained from a feasible solution in $\mathtt{J}$ is a matching.

***Matching Constraints.*** The feasibility of a matching can be ensured with the following three sets of constraints.

$$\sum_{p_j \in A_i} x_{i,j} \leq 1 \qquad (1 \leq i \leq n_1), \tag{1}$$

$$\sum_{i=1}^{n_1} x_{i,j} \leq c_j \qquad (1 \leq j \leq n_2), \tag{2}$$

$$\sum_{i=1}^{n_1} \sum_{p_j \in P_k} x_{i,j} \leq d_k \qquad (1 \leq k \leq n_3)\ . \tag{3}$$

Note that (1) implies that each student $s_i \in \mathcal{S}$ is not assigned to more than one project, while (2) and (3) implies that the capacity of each project $p_j \in \mathcal{P}$ and each lecturer $l_k \in \mathcal{L}$ is not exceeded.

We define $rank(s_i, p_j)$, the *rank* of $p_j$ on $s_i$'s preference list, to be $r+1$ where $r$ is the number of projects that $s_i$ prefers to $p_j$. An analogous definition holds for $rank(l_k, p_j)$, the *rank* of $p_j$ on $l_k$'s preference list. With respect to an acceptable pair $(s_i, p_j)$, we define $S_{i,j} = \{p_{j'} \in A_i : rank(s_i, p_{j'}) \leq rank(s_i, p_j)\}$, the set of projects that $s_i$ likes as much as $p_j$. For a project $p_j$ offered by lecturer $l_k \in \mathcal{L}$, we also define $T_{k,j} = \{p_q \in P_k : rank(l_k, p_j) < rank(l_k, p_q)\}$, the set of projects that are worse than $p_j$ on $l_k$'s preference list.

In what follows, we fix an arbitrary acceptable pair $(s_i, p_j)$ and we impose constraints to ensure that $(s_i, p_j)$ is not a blocking pair of the matching $M$ (that is, $(s_i, p_j)$ is not a type 1(a), type 1(b) or type 1(c) blocking pair of $M$). Firstly, let $l_k$ be the lecturer who offers $p_j$.

***Blocking Pair Constraints.*** We define $\theta_{i,j} = 1 - \sum_{p_{j'} \in S_{i,j}} x_{i,j'}$. Intuitively, $\theta_{i,j} = 1$ if and only if $s_i$ is unassigned in $M$ or prefers $p_j$ to $M(s_i)$. Next we create a binary variable $\alpha_j$ in J such that $\alpha_j = 1$ corresponds to the case when $p_j$ is undersubscribed in $M$. We enforce this condition by imposing the following constraint.

$$c_j \alpha_j \geq c_j - \sum_{i'=1}^{n_1} x_{i',j} \ , \tag{4}$$

where $\sum_{i'=1}^{n_1} x_{i',j} = |M(p_j)|$. If $p_j$ is undersubscribed in $M$ then the RHS of (4) is at least 1, and this implies that $\alpha_j = 1$. Otherwise, $\alpha_j$ is not constrained. Now let $\gamma_{i,j,k} = \sum_{p_{j'} \in T_{k,j}} x_{i,j'}$. Intuitively, if $\gamma_{i,j,k} = 1$ in $S$ then $s_i$ is assigned to a project $p_{j'}$ offered by $l_k$ in $M$, where $l_k$ prefers $p_j$ to $p_{j'}$. The following constraint ensures that $(s_i, p_j)$ does not form a type 1(a) blocking pair of $M$.

$$\boxed{\theta_{i,j} + \alpha_j + \gamma_{i,j,k} \leq 2 \ .} \tag{5}$$

Note that if the sum of the binary variables in the LHS of (5) is less than or equal to 2, this implies that at least one of the variables, say $\gamma_{i,j,k}$, is 0. Thus the pair $(s_i, p_j)$ is not a type 1(a) blocking pair of $M$.

Next we define $\beta_{i,k} = \sum_{p_{j'} \in P_k} x_{i,j'}$. Clearly, $s_i$ is assigned to a project offered by $l_k$ in $M$ if and only if $\beta_{i,k} = 1$ in $S$. Now we create a binary variable $\delta_k$ in J such that $\delta_k = 1$ in $S$ corresponds to the case when $l_k$ is undersubscribed in $M$. We enforce this condition by imposing the following constraint.

$$d_k \delta_k \geq d_k - \sum_{i'=1}^{n_1} \sum_{p_{j'} \in P_k} x_{i',j'} \ , \tag{6}$$

where $\sum_{i'=1}^{n_1} \sum_{p_{j'} \in P_k} x_{i',j'} = |M(l_k)|$. If $l_k$ is undersubscribed in $M$ then the RHS of (6) is at least 1, and this implies that $\delta_k = 1$. Otherwise, $\delta_k$ is not constrained. The following constraint ensures that $(s_i, p_j)$ does not form a type 1(b) blocking pair of $M$.

$$\boxed{\theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \delta_k \leq 3 \ .} \tag{7}$$

We define $D_{k,j} = \{p_{j'} \in P_k : rank(l_k, p_{j'}) \leq rank(l_k, p_j)\}$, the set of projects that $l_k$ likes as much as $p_j$. Next, we create a binary variable $\eta_{j,k}$ in J such that $\eta_{j,k} = 1$ if $l_k$ is full and prefers $p_j$ to his worst non-empty project in $S$. We enforce this by imposing the following constraint.

$$d_k \eta_{j,k} \geq d_k - \sum_{i'=1}^{n_1} \sum_{p_{j'} \in D_{k,j}} x_{i',j'} \ . \tag{8}$$

Finally, to avoid a type 1(c) blocking pair, we impose the following constraint.

$$\boxed{\theta_{i,j} + \alpha_j + (1 - \beta_{i,k}) + \eta_{j,k} \leq 3 \ .} \tag{9}$$

Next, we give the constraints to ensure that the matching obtained from a feasible solution in J is coalition-free.

***Coalition Constraints.*** First, we introduce some additional notation. Given an instance $\mathtt{I}'$ of SPA-P and a matching $M'$ in $\mathtt{I}'$, we define the *envy graph* $G(M') = (\mathcal{S}, A)$, where the vertex set $\mathcal{S}$ is the set of students in $\mathtt{I}'$, and the arc set $A = \{(s_i, s_{i'}) : s_i \text{ prefers } M'(s_{i'}) \text{ to } M'(s_i)\}$. It is clear that the matching $M_2 = \{(s_1, p_1), (s_2, p_2), (s_3, p_3)\}$ admits a coalition $\{s_1, s_2\}$ with respect to the instance given in Fig. 1. The resulting envy graph $G(M_2)$ is illustrated below.

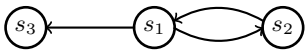


**Fig. 2.** The envy graph $G(M_2)$ corresponding to the SPA-P instance in Fig. 1.

Clearly, $G(M')$ contains a directed cycle if and only if $M'$ admits a coalition. Moreover, $G(M')$ is acyclic if and only if it admits a topological ordering. Now to ensure that the matching $M$ obtained from a feasible solution $S$ under $\mathtt{J}$ is coalition-free, we will enforce $\mathtt{J}$ to encode the envy graph $G(M)$ and impose the condition that it must admit a topological ordering. In what follows, we build on our IP model $\mathtt{J}$ of $\mathtt{I}$.

We create a binary variable $e_{i,i'}$ for each $(s_i, s_{i'}) \in \mathcal{S} \times \mathcal{S}$, $s_i \neq s_{i'}$, such that the $e_{i,i'}$ variables will correspond to the adjacency matrix of $G(M)$. For each $i$ and $i'$ $(1 \leq i \leq n_1,\ 1 \leq i' \leq n_1,\ i \neq i')$ and for each $j$ and $j'$ $(1 \leq j \leq n_2,\ 1 \leq j' \leq n_2)$ such that $s_i$ prefers $p_{j'}$ to $p_j$, we impose the following constraint.

$$e_{i,i'} + 1 \geq x_{i,j} + x_{i',j'} \ . \tag{10}$$

If $(s_i, p_j) \in M$ and $(s_{i'}, p_{j'}) \in M$ and $s_i$ prefers $p_{j'}$ to $p_j$, then $e_{i,i'} = 1$ and we say $s_i$ *envies* $s_{i'}$. Otherwise, $e_{i,i'}$ is not constrained. Next we enforce the condition that $G(M)$ must have a topological ordering. To hold the label of each vertex in a topological ordering, we create an integer-valued variable $v_i$ corresponding to each student $s_i \in \mathcal{S}$ (and intuitively to each vertex in $G(M)$). We wish to enforce the constraint that if $e_{i,i'} = 1$ (that is, $(s_i, s_{i'}) \in A$), then $v_i < v_{i'}$ (that is, the label of vertex $s_i$ is smaller than the label of vertex $s_{i'}$). This is achieved by imposing the following constraint for all $i$ and $i'$ $(1 \leq i \leq n_1,\ 1 \leq i' \leq n_1,\ i \neq i')$.

$$\boxed{v_i < v_{i'} + n_1(1 - e_{i,i'}) \ .} \tag{11}$$

Note that the LHS of (11) is strictly less than the RHS of (11) if and only if $G(M)$ does not admit a directed cycle, and this implies that $M$ is coalition-free.

***Variables.*** We define a collective notation for each variable involved in $\mathtt{J}$ as follows.

$$\begin{aligned} X &= \{x_{i,j} : 1 \leq i \leq n_1, 1 \leq j \leq n_2\}, \quad \Lambda = \{\alpha_j : 1 \leq j \leq n_2\}, \\ H &= \{\eta_{j,k} : 1 \leq j \leq n_2, 1 \leq k \leq n_3\}, \quad \Delta = \{\delta_k : 1 \leq k \leq n_3\}, \\ E &= \{e_{i,i'} : 1 \leq i \leq n_1, 1 \leq i' \leq n_1\}, \quad V = \{v_i : 1 \leq i \leq n_1\} \ . \end{aligned}$$

***Objective Function.*** The objective function given below is a summation of all the $x_{i,j}$ binary variables. It seeks to maximize the number of students assigned (that is, the cardinality of the matching).

$$\boxed{\max \sum_{i=1}^{n_1} \sum_{p_j \in A_i} x_{i,j} \ .} \tag{12}$$

Finally, we have constructed an IP model J of I comprising the set of integer-valued variables $X, \Lambda, H, \Delta, E$ and $V$, the set of constraints (1) - (11) and an objective function (12). Note that J can then be used to solve MAX-SPA-P optimally. Given an instance I of SPA-P formulated as an IP model J using the above transformation, we present the following result regarding the correctness of J (see [13] for proof).

**Theorem 1.** *A feasible solution to* J *is optimal if and only if the corresponding stable matching in* I *is of maximum cardinality.*

## 4 Empirical Analysis

In this section we present results from an empirical analysis that investigates how the sizes of the stable matchings produced by the approximation algorithms compares to the optimal solution obtained from our IP model, on SPA-P instances that are both randomly-generated and derived from real datasets.

### 4.1 Experimental Setup

There are clearly several parameters that can be varied, such as the number of students, projects and lecturers; the length of the students' preference lists; as well as the total capacities of the projects and lecturers. For each range of values for the first two parameters, we generated a set of random SPA-P instances. In each set, we record the average size of a stable matching obtained from running the approximation algorithms and the IP model. Further, we consider the average time taken for the IP model to find an optimal solution.

By design, the approximation algorithms were randomised with respect to the sequence in which students apply to projects, and the choice of students to reject when projects and/or lecturers become full. In the light of this, for each dataset, we also run the approximation algorithms 100 times and record the size of the largest stable matching obtained over these runs. Our experiments therefore involve five algorithms: the optimal IP-based algorithm, the two approximation algorithms run once, and the two approximation algorithms run 100 times.

We performed our experiments on a machine with dual Intel Xeon CPU E5-2640 processors with 64GB of RAM, running Ubuntu 14.04. Each of the approximation algorithms was implemented in Java[1]. For our IP model, we carried

[1] https://github.com/sofiat-olaosebikan/spa-p-isco-2018

out the implementation using the Gurobi optimisation solver in Java[1]. For correctness testing on these implementations, we designed a stability checker which verifies that the matching returned by the approximation algorithms and the IP model does not admit a blocking pair or a coalition.

### 4.2 Experimental Results

**Randomly-generated Datasets.** All the SPA-P instances we randomly generated involved $n_1$ students ($n_1$ is henceforth referred to as the size of the instance), $0.5n_1$ projects, $0.2n_1$ lecturers and $1.1n_1$ total project capacity which was randomly distributed amongst the projects. The capacity for each lecturer $l_k$ was chosen randomly to lie between the highest capacity of the projects offered by $l_k$ and the sum of the capacities of the projects that $l_k$ offers. In the first experiment, we present results obtained from comparing the performance of the IP model, with and without the coalition constraints in place.

***Experiment 0.*** We increased the number of students $n_1$ while maintaining a ratio of projects, lecturers, project capacities and lecturer capacities as described above. For various values of $n_1$ ($100 \leq n_1 \leq 1000$) in increments of 100, we created 100 randomly-generated instances. Each student's preference list contained a minimum of 2 and a maximum of 5 projects. With respect to each value of $n_1$, we obtained the average time taken for the IP solver to output a solution, both with and without the coalition constraints being enforced. The results, displayed in Table 1, show that when we removed the coalition constraints, the average time for the IP solver to output a solution is significantly faster than when we enforced the coalition constraints.

In the remaining experiments, we thus remove the constraints that enforce the absence of a coalition in the solution. We are able to do this for the purposes of these experiments because the largest size of a stable matching is equal to the largest size of a matching that potentially admits a coalition but admits no blocking pair[2], and we were primarily concerned with measuring stable matching cardinalities. However the absence of the coalition constraints should be borne in mind when interpreting the IP solver runtime data in what follows.

In the next two experiments, we discuss results obtained from running the five algorithms on randomly-generated datasets.

***Experiment 1.*** As in the previous experiment, we maintained the ratio of the number of students to projects, lecturers and total project capacity; as well as the length of the students' preference lists. For various values of $n_1$ ($100 \leq n_1 \leq 2500$) in increments of 100, we created 1000 randomly-generated instances. With respect to each value of $n_1$, we obtained the average sizes of stable matchings constructed by the five algorithms run over the 1000 instances. The result displayed in Fig. 3 (and also in Fig. 4) shows the ratio of the average size of the

[2] This holds because the number of students assigned to each project and lecturer in the matching remains the same even after the students involved in such coalition permute their assigned projects.

stable matching produced by the approximation algorithms with respect to the maximum cardinality matching produced by the IP solver.

Figure 3 shows that each of the approximation algorithms produces stable matchings with a much higher cardinality from multiple runs, compared to running them only once. Also, the average time taken for the IP solver to find a maximum cardinality matching increases as the size of the instance increases, with a running time of less than one second for instance size 100, increasing roughly linearly to 13 seconds for instance size 2500 (see [13, Fig. 3(b)]).

***Experiment 2.*** In this experiment, we varied the length of each student's preference list while maintaining a fixed number of students, projects, lecturers and total project capacity. For various values of $x$ ($2 \leq x \leq 10$), we generated 1000 instances, each involving 1000 students, with each student's preference list containing exactly $x$ projects. The result for all values of $x$ is displayed in Fig. 4. Figure 4 shows that as we increase the preference list length, the stable matchings produced by each of the approximation algorithms gets close to having maximum cardinality. It also shows that with a preference list length greater than 5, the $\frac{3}{2}$-approximation algorithm produces an optimal solution, even on a single run. Moreover, the average time taken for the IP solver to find a maximum matching increases as the length of the students' preference lists increases, with a running time of two seconds when each student's preference list is of length 2, increasing roughly linearly to 17 seconds when each student's preference list is of length 10 (see [13, Fig. 4(b)]).

**Real Datasets.** The real datasets in this paper are based on actual student preference data and manufactured lecturer data from previous runs of student-project allocation processes at the School of Computing Science, University of Glasgow. Table 2 shows the properties of the real datasets, where $n_1, n_2$ and $n_3$ denotes the number of students, projects and lecturers respectively; and $l$ denotes the length of each student's preference list. For all these datasets, each project has a capacity of 1. In the next experiment, we discuss how the lecturer preferences were generated. We also discuss the results obtained from running the five algorithms on the corresponding SPA-P instances.

***Experiment 3.*** We derived the lecturer preference data from the real datasets as follows. For each lecturer $l_k$, and for each project $p_j$ offered by $l_k$, we obtained the number $a_j$ of students that find $p_j$ acceptable. Next, we generated a strict preference list for $l_k$ by arranging $l_k$'s proposed projects in (i) a random manner, (ii) ascending order of $a_j$, and (iii) descending order of $a_j$, where (ii) and (iii) are taken over all projects that $l_k$ offers. Table 2 shows the size of stable matchings obtained from the five algorithms, where $A, B, C, D$ and $E$ denotes the solution obtained from the IP model, 100 runs of $\frac{3}{2}$-approximation algorithm, single run of $\frac{3}{2}$-approximation algorithm, 100 runs of 2-approximation algorithm, and single run of 2-approximation algorithm respectively. The results are essentially consistent with the findings in the previous experiments, that is, the $\frac{3}{2}$-approximation algorithm produces stable matchings whose sizes are close to optimal.

**Table 1.** Results for Experiment 0. Average time (in seconds) for the IP solver to output a solution, both with and without the coalition constraints being enforced.

| Size of instance | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| Av. time without coalition | 0.12 | 0.27 | 0.46 | 0.69 | 0.89 | 1.17 | 1.50 | 1.86 | 2.20 | 2.61 |
| Av. time with coalition | 0.71 | 2.43 | 4.84 | 9.15 | 13.15 | 19.34 | 28.36 | 38.18 | 48.48 | 63.50 |

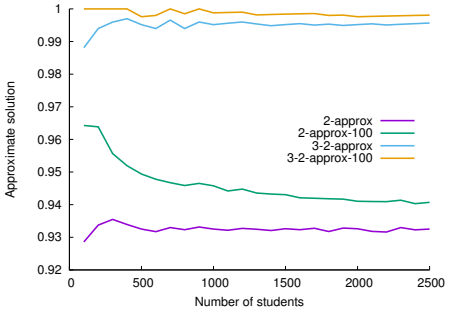


**Fig. 3.** Result for Experiment 1.

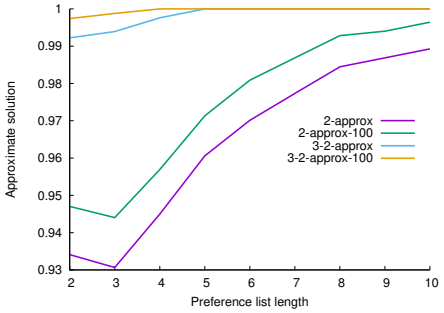


**Fig. 4.** Result for Experiment 2.

**Table 2.** Properties of the real datasets and results for Experiment 3.

| | | | | | Random | | | | | Most popular | | | | | Least popular | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Year | $n_1$ | $n_2$ | $n_3$ | $l$ | $A$ | $B$ | $C$ | $D$ | $E$ | $A$ | $B$ | $C$ | $D$ | $E$ | $A$ | $B$ | $C$ | $D$ | $E$ |
| 2014 | 55 | 149 | 38 | 6 | 55 | 55 | 55 | 54 | 53 | 55 | 55 | 55 | 54 | 50 | 55 | 55 | 55 | 54 | 52 |
| 2015 | 76 | 197 | 46 | 6 | 76 | 76 | 76 | 76 | 72 | 76 | 76 | 76 | 76 | 72 | 76 | 76 | 76 | 76 | 75 |
| 2016 | 92 | 214 | 44 | 6 | 84 | 82 | 83 | 77 | 75 | 85 | 85 | 83 | 79 | 76 | 82 | 80 | 77 | 76 | 74 |
| 2017 | 90 | 289 | 59 | 4 | 89 | 87 | 85 | 80 | 76 | 90 | 89 | 86 | 81 | 79 | 88 | 85 | 84 | 80 | 77 |

### 4.3 Discussions and Concluding Remarks

The results presented in this section suggest that even as we increase the number of students, projects, lecturers, and the length of the students' preference lists, each of the approximation algorithms finds stable matchings that are close to having maximum cardinality, outperforming their approximation factor. Perhaps most interesting is the $\frac{3}{2}$-approximation algorithm, which finds stable matchings that are very close in size to optimal, even on a single run. These results also holds analogously for the instances derived from real datasets.

We remark that when we removed the coalition constraints, we were able to run the IP model on an instance size of 10000, with the solver returning a maximum matching in an average time of 100 seconds, over 100 randomly-generated instances. This shows that the IP model (without enforcing the coalition constraints), can be run on SPA-P instances that appear in practice, to find maximum cardinality matchings that admit no blocking pair. Coalitions should then

be eliminated in polynomial time by repeatedly constructing an *envy graph*, similar to the one described in [11, p.290], finding a directed cycle and letting the students in the cycle swap projects.

## 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):79–91, 2007.
2. A.A. Anwar and A.S. Bahaj. Student project allocation using integer programming. *IEEE Transactions on Education*, 46(3):359–367, 2003.
3. R. Calvo-Serrano, G. Guillén-Gosálbez, S. Kohn, and A. Masters. Mathematical programming approach for optimally allocating students' projects to academics in large cohorts. *Education for Chemical Engineers*, 20:11–21, 2017.
4. M. Chiarandini, R. Fagerberg, and S. Gualandi. Handling preferences in student-project allocation. *Annals of Operations Research,* to appear, 2018.
5. D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.
6. P.R. Harper, V. de Senna, I.T. Vieira, and A.K. Shahani. A genetic algorithm for the project assignment problem. *Computers and Operations Research*, 32:1255–1265, 2005.
7. K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation bounds for the student-project allocation problem with preferences over projects. *Journal of Discrete Algorithms*, 13:59–66, 2012.
8. D. Kazakov. Co-ordination of student-project allocation. Manuscript, University of York, Department of Computer Science. Available from `http://www-users.cs.york.ac.uk/kazakov/papers/proj.pdf`, 2001 (last accessed 8 March 2018).
9. Z. Király. Better and simpler approximation algorithms for the stable marriage problem. *Algorithmica*, 60:3–20, 2011.
10. A. Kwanashie, R.W. Irving, D.F. Manlove, and C.T.S. Sng. Profile-based optimal matchings in the Student–Project Allocation problem. In *Proceedings of IWOCA '14*, vol. 8986 of *Lecture Notes in Computer Science*, pp. 213–225. Springer, 2015.
11. D.F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.
12. D.F. Manlove and G. O'Malley. Student project allocation with preferences over projects. *Journal of Discrete Algorithms*, 6:553–560, 2008.
13. D.F. Manlove, D. Milne, and S. Olaosebikan. An Integer Programming Approach to the Student-Project Allocation Problem with Preferences over Projects. CoRR abs/1804.09993 (2018). Available from `https://arxiv.org/abs/1804.09993`.
14. L.G. Proll. A simple method of assigning projects to students. *Operational Research Quarterly*, 23(2):195–201, 1972.
15. A.E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984.
16. C.Y. Teo and D.J. Ho. A systematic approach to the implementation of final year project in an electrical engineering undergraduate course. *IEEE Transactions on Education*, 41(1):25–30, 1998.
17. `http://www.gurobi.com` (Gurobi Optimization website). Accessed 09-01-2018.
18. `https://www.gnu.org/software/glpk` (GNU Linear Proramming Kit). Accessed 09-01-2018.
19. `http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/` (CPLEX Optimization Studio). Accessed 19-05-2017.