Student-Project Allocation with Preferences over Projects*

David F. Manlove[†,‡] and Gregg O'Malley[†]

Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK.
Email: {davidm,gregg}@dcs.gla.ac.uk.

Abstract

We study the problem of allocating students to projects, where both students and lecturers have preferences over projects, and both projects and lecturers have capacities. In this context we seek a *stable matching* of students to projects, which respects these preference and capacity constraints. Here, the stability definition generalises the corresponding notion in the context of the classical Hospitals / Residents problem. We show that stable matchings can have different sizes, which motivates MAX-SPA-P, the problem of finding a maximum cardinality stable matching. We prove that MAX-SPA-P is NP-hard and not approximable within δ, for some $\delta > 1$, unless P=NP. On the other hand, we give an approximation algorithm with a performance guarantee of 2 for MAX-SPA-P.

Keywords: Matching problem; Stable matching; NP-hardness; Approximation hardness; Approximation algorithm

1 Introduction

As part of the senior level of many undergraduate degree courses, students are required to undertake some form of project work. Typically the available projects are advertised to the students, and having browsed through the descriptions, each student (either explicitly or implicitly) forms a preference list over the projects that he/she finds acceptable. Lecturers may also have preferences over the students and/or the projects that they offer. There may also be upper bounds on the number of students that can be assigned to a particular project, and the number of students that a given lecturer is willing to supervise.

We refer to the problem of assigning students to projects subject to these preference lists and capacity constraints as the *Student-Project Allocation problem* (SPA).

*A preliminary version of this paper appeared in *Proceedings of ACiD 2005: the 1st Algorithms and Complexity in Durham workshop*, volume 4 of *Texts in Algorithmics*, pages 69-80, KCL Publications, 2005.

[†]Supported by Engineering and Physical Sciences Research Council grant GR/R84597/01.

[‡]Supported by Royal Society of Edinburgh/Scottish Executive Personal Research Fellowship.

Given the large numbers of students that are typically involved in such applications, there is a growing interest in automating the process of allocating students to projects using centralised matching schemes that incorporate efficient algorithms for SPA. Examples of such automated systems are in use at the Department of Computer Science, University of York [4, 9, 13], the University of Southampton [3, 8] and elsewhere [12].

SPA is a generalisation of the classical *Hospitals / Residents* problem (HR) [5, 6] which has applications to the annual match of graduating medical students (or *residents*) to their first hospital posts in a number of countries [11]. In the US, for example, the National Resident Matching Program (NRMP) deals with the allocation of some 31,000 medical students annually. The NRMP utilises an algorithm that essentially solves an extension of HR, forming a *stable matching* of residents to hospitals, taking into account hospital capacities, and the preferences of residents over hospitals and vice versa. Informally, a *matching* guarantees that no resident is assigned to more than one hospital, and no hospital is assigned more residents than its capacity, whilst the concept of *stability* ensures that no resident and hospital who are not matched together would rather be assigned to each other than remain with their current assignees. Such a pair could improve their situations by coming to a private arrangement outside of the matching, undermining its integrity. It has been convincingly argued [11] that, when preference lists exist on both sides of a two-sided matching market (for example involving residents and hospitals, or students and lecturers), the key property that a matching should satisfy is that of stability.

Stable matchings in the context of SPA have been considered previously. In [1], a model for SPA was introduced in which students have preferences over projects, whilst lecturers have preferences over students. A linear-time algorithm for finding a stable matching of students to projects in this context was described, in terms of a stability definition that is a natural generalisation of stability in the context of HR. This algorithm constructs the *student-optimal* stable matching, in which each student obtains the best project that he/she could obtain in any stable matching. A second linear-time algorithm [2] finds the *lecturer-optimal* stable matching, in which each lecturer obtains the best (in a precise sense) set of students that he/she could obtain in any stable matching.

In some cases, neither lecturers nor students find it desirable that lecturers should form preference lists over students. For example, if such lists are derived largely on the basis of academic merit, then students who have performed poorly in previous examinations are less likely to be assigned to preferable projects if these projects are popular, and could therefore struggle to improve their academic performance. However, often it is the case that lecturers have tangible preferences over the projects that they offer. For example, a lecturer may strongly prefer to supervise a particular project if it is closely connected with his research. In this paper we consider the variant of SPA in which lecturers rank in strict order of preference the projects that they offer. Under this condition, implicitly each lecturer is indifferent among those students who find acceptable a given project that he/she offers.

Our contribution is as follows. In Section 2 we give a formal definition of the variant of SPA in which lecturers have preferences over projects, which we refer to as SPA-P, formulating an appropriate stability definition in this context. A stable

Student preferences	Lecturer preferences
$s_1 :\ p_1\ \ p_2$	$l_1 :\ p_1$
$s_2 :\ p_1$	$l_2 :\ p_2$
	$c_1 = c_2 = d_1 = d_2 = 1$

Figure 1: An instance I_1 of SPA-P.

matching M guarantees that (i) no student and lecturer could improve relative to M by forming a private arrangement involving some project, and (ii) no coalition of students could permute their assigned projects in M so as to improve their allocation. We show that, in a given instance of SPA-P, stable matchings can have different sizes. In most practical situations we seek to allocate as many students to projects as possible, and this motivates the problem of finding a maximum cardinality stable matching (henceforth a maximum stable matching). In Section 3 we show that this problem is NP-hard and not approximable within δ, for some $\delta > 1$, unless P=NP. The result holds even in the special case that each project and lecturer can accommodate only one student, and each person's preference list is of bounded length. However in Section 4, we give an approximation algorithm for the problem that admits a performance guarantee of 2. This algorithm also demonstrates that every instance of SPA-P admits at least one stable matching. Finally, Section 5 contains some concluding remarks.

We remark that SPA-P is an example of a matching problem in which the members of two sets of entities (namely the students and lecturers) each have preferences over the members of a common third entity (namely the projects). As far as we are aware, SPA-P is the first matching problem of this type to be considered in the literature. The previous formulations of SPA to have been considered either do not permit lecturer preferences [10, 12, 3, 8] (so stability is not relevant in these contexts) or involve lecturer preferences over students [4, 9, 1, 13, 2].

2 Definition of SPA-P

We begin by defining an instance of SPA-P, the Student-Project Allocation problem with preferences over Projects. An instance of SPA-P involves a set $\mathcal{S}$ of *students*, a set $\mathcal{P}$ of *projects*, and a set $\mathcal{L}$ of *lecturers*. Each lecturer $l_k \in \mathcal{L}$ *offers* a set of projects, denoted by P_k. We assume that $P_1, \ldots, P_q$ partitions $\mathcal{P}$, where $q = |\mathcal{L}|$, so that each project is offered by a unique lecturer. Also, each student $s_i \in \mathcal{S}$ has an *acceptable* set of projects $A_i \subseteq \mathcal{P}$. Moreover s_i ranks A_i in strict order of preference. Similarly 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 example SPA-P instance with $\mathcal{S} = \{s_1, s_2\}$, $\mathcal{P} = \{p_1, p_2\}$ and $\mathcal{L} = \{l_1, l_2\}$, where $A_1 = \{p_1, p_2\}$, $A_2 = \{p_1\}$, $P_1 = \{p_1\}$ and $P_2 = \{p_2\}$, is shown in Figure 1.

An *assignment* M is a subset of $\mathcal{S} \times \mathcal{P}$ such that $(s_i, p_j) \in M$ implies that $p_j \in A_i$ (i.e. s_i finds p_j acceptable). If $(s_i, p_j) \in M$, we say that s_i is *assigned to* p_j, and p_j is *assigned* s_i. For ease of exposition, if s_i is assigned to p_j and l_k is the lecturer

who offers p_j, we may also say that s_i *is assigned to* l_k, and l_k *is assigned* s_i.

For each $r \in \mathcal{S} \cup \mathcal{P} \cup \mathcal{L}$, we denote by $M(r)$ the set of assignees of r in M. If $s_i \in \mathcal{S}$ and $M(s_i) = \emptyset$, we say that s_i is *unassigned*, otherwise s_i is *assigned*. Similarly, any project $p_j \in \mathcal{P}$ is *under-subscribed*, *full* or *over-subscribed* according as $|M(p_j)|$ is less than, equal to, or greater than c_j, respectively. The same three terms are defined for a lecturer $l_k \in \mathcal{L}$ with respect to l_k's capacity d_k. A project $p_j \in \mathcal{P}$ is said to be *non-empty* if $|M(p_j)| > 0$. Similarly a lecturer $l_k \in \mathcal{L}$ is said to be *non-empty* if $|M(l_k)| > 0$.

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 $l_k \in \mathcal{L}$ (i.e. each student is assigned to at most one project, and no project or lecturer is over-subscribed). For notational convenience, given a matching M and a student $s_i \in \mathcal{S}$ such that $M(s_i) \neq \emptyset$, where there is no ambiguity the notation $M(s_i)$ is also used to refer to the single member of $M(s_i)$.

A (student,project) pair $(s_i, p_j) \in (\mathcal{S} \times \mathcal{P}) \backslash M$ *blocks* a matching M, or is a *blocking pair* of M, if the following conditions are satisfied relative to M:

1. $p_j \in A_i$ (i.e. s_i finds p_j acceptable);
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) $s_i \in M(l_k)$ and l_k prefers p_j to $M(s_i)$, or
   (b) $s_i \notin M(l_k)$ and l_k is under-subscribed, or
   (c) $s_i \notin M(l_k)$ and l_k is full and l_k prefers p_j to his worst non-empty project,

   where l_k is the lecturer who offers p_j.

We now give some intuition for the definition of a blocking pair. Suppose that (s_i, p_j) forms a blocking pair with respect to matching M, and let l_k be the lecturer who offers p_j.

We assume that s_i prefers to be assigned to an acceptable project p_j rather than remain unassigned, so Condition 2 indicates how a student could improve relative to M. We now consider Condition 3. If p_j is already full, then l_k would not improve by rejecting a student assigned to p_j and taking on s_i instead (recall that l_k is indifferent among those students who find p_j acceptable). Thus p_j must be under-subscribed. Firstly suppose that s_i was already assigned to a project p_r offered by l_k. In this case l_k would only let s_i change projects from p_r to p_j if he prefers p_j to p_r – Condition 3(a). Secondly suppose that s_i was not already assigned to a project offered by l_k. If l_k is under-subscribed then both p_j and l_k have a free place for s_i – Condition 3(b). Otherwise if l_k is full and l_k prefers p_j to his worst non-empty project p_r, then l_k could improve by rejecting a student from p_r and taking on s_i to do p_j instead – Condition 3(c).

A blocking pair thus gives a situation in which a given matching M could be undermined. Another way in which this could occur is through the existence of a *coalition*. A *coalition* is a set of students $\{s_{i_0}, \ldots, s_{i_{r-1}}\}$, for some $r \geq 2$, each of whom is assigned in M, such that s_{i_j} prefers $M(s_{i_{j+1}})$ to $M(s_{i_j})$ $(0 \leq j \leq r-1$, where addition is taken modulo $r)$. That is, the students in the coalition could permute

Student preferences	Lecturer preferences
$s_1 :\ p_2\ \ p_1$	$l_1 :\ p_1\ \ p_2$
$s_2 :\ p_1\ \ p_2$	
	$c_1 = c_2 = 1; d_1 = 2$

Figure 2: An instance I_2 of SPA-P.

the projects that they have been assigned to in M so as to be better off. Notice that, were such a permutation of projects to occur, the number of students assigned to each project and lecturer would not change. Moreover, since each lecturer l_k is implicitly indifferent between those students who find acceptable a given project offered by l_k, the lecturers involved in the permutation have no explicit incentive to prevent the switch from occurring. Figure 2 gives a simple instance of SPA-P in which the matching $M = \{(s_1, p_1), (s_2, p_2)\}$ admits no blocking pair but does admit a coalition, namely $\{s_1, s_2\}$. Define a matching to be *coalition-free* if it admits no coalition. We remark that, in the context of HR, or SPA variants involving lecturer preferences over students, a matching that admits no blocking pair cannot admit a coalition of students (or residents), since the lecturers (or hospitals) involved would, by definition of a blocking pair, be worse off were the switch to occur, and hence would not, in practice, agree to such a switch.

Define a matching to be *stable* if it admits no blocking pair and is coalition-free. It turns out that, with respect to this definition, for a given instance of SPA-P, stable matchings can have different sizes, as the example instance I_1 shown in Figure 1 illustrates. It may be verified that each of the matchings $M_1 = \{(s_1, p_1)\}$ and $M_2 = \{(s_1, p_2), (s_2, p_1)\}$ is stable in I_1. In practical situations, often a key priority is to match as many students to acceptable projects as possible, so this naturally leads one to consider the complexity of finding a maximum stable matching, given a SPA-P instance.

3 Hardness of approximating a maximum stable matching

Given an instance I of SPA-P, let $s^+(I)$ denote the maximum size of a stable matching in I. Define MAX-SPA-P to be the problem of computing $s^+(I)$, given an instance I of SPA-P. In this section we show that MAX-SPA-P is NP-hard, and moreover that there exists a constant $\delta_1 > 1$ such that it is NP-hard to approximate MAX-SPA-P within δ_1. The result holds even if each project and lecturer has capacity 1, and all preference lists are of bounded length.

We prove this result using a reduction from a problem relating to matchings in graphs. A matching M in a graph G is said to be *maximal* if no proper superset of M is a matching in G. Let $\beta_1^-(G)$ denote the minimum size of a maximal matching in G. Define MIN-MM to be the problem of computing $\beta_1^-(G)$, given a graph G. The following result regarding the inapproximability of MIN-MM is proved in [7].

Theorem 1. *Let $G = (V, E)$ be an instance of* MIN-MM*, where $m = |E|$. Then there exist constants $c_0 > 0$ and $\delta_0 > 1$ such that it is NP-hard to distinguish between the cases $\beta_1^-(G) \leq c_0 m$ and $\beta_1^-(G) > \delta_0 c_0 m$. Hence it is NP-hard to approximate* MIN-MM *within δ_0. The result holds even for subdivision graphs*[1] *of cubic graphs.*

We will use Theorem 1 together with the notion of a *gap-preserving reduction* [14, p.308], which may be defined as follows.

Definition 2. *Let Π_1 be a minimisation problem and let Π_2 be a maximisation problem. Denote by $OPT_i(x)$ the optimal measure over all feasible solutions for a given instance x of Π_i ($i \in \{1, 2\}$). Let $\alpha \geq 1$ be some constant and let g_1 be a function that maps an instance x of Π_1 to a positive rational number. Then a* gap-preserving reduction *from Π_1 to Π_2 is a tuple $\langle f, \beta, g_2 \rangle$ such that:*

- *f maps an instance x of Π_1 to an instance $f(x)$ of Π_2 in polynomial time;*
- *$\beta \geq 1$ is a constant;*
- *g_2 maps an instance $f(x)$ of Π_2 to a positive rational number;*
- *for any instance x of Π_1:*
  - *if $OPT_1(x) \leq g_1(x)$, then $OPT_2(f(x)) \geq g_2(f(x))$;*
  - *if $OPT_1(x) > \alpha g_1(x)$, then $OPT_2(f(x)) < (1/\beta) g_2(f(x))$.*

The following proposition is an immediate consequence of Definition 2.

Proposition 3. *Let Π_1 be a minimisation problem and let Π_2 be a maximisation problem, and suppose that there is a gap-preserving reduction from Π_1 to Π_2. Assuming the notation of Definition 2, suppose further that it is NP-hard to distinguish between instances x of Π_1 such that $OPT_1(x) \leq g_1(x)$ and $OPT_1(x) > \alpha g_1(x)$. Then it is NP-hard to distinguish between instances $f(x)$ of Π_2 such that $OPT_2(f(x)) \geq g_2(f(x))$ and $OPT_2(f(x)) < (1/\beta) g_2(f(x))$. Hence it is NP-hard to approximate Π_2 within β.*

We use Proposition 3, together with Theorem 1, to prove the NP-hardness and inapproximability result for MAX-SPA-P.

Theorem 4. MAX-SPA-P *is NP-hard. Moreover it is NP-hard to approximate* MAX-SPA-P *within δ_1, for some $\delta_1 > 1$. The result holds even if each project and lecturer has capacity 1, and all preference lists are of bounded length.*

Proof. Let G (a subdivision graph of some cubic graph G') be an instance of MIN-MM. Then G is a bipartite graph, so that $G = (U, W, E)$, where without loss of generality each vertex in U has degree 2 and each vertex in W has degree 3. Suppose that $n_1 = |U|$ and $n_2 = |W|$. Let $U = \{u_1, u_2, \ldots, u_{n_1}\}$ and $W = \{w_1, w_2, \ldots, w_{n_2}\}$. For each $u_i \in U$, let w_{j_i} and w_{k_i} be the two neighbours of u_i in G, where $j_i < k_i$.

[1] Given a graph G, the *subdivision graph* of G, denoted by $S(G)$, is obtained by subdividing each edge $\{u, w\}$ of G in order to obtain two edges $\{u, v\}$ and $\{v, w\}$ of $S(G)$, where v is a new vertex.

$$\text{Student preferences:} \quad \begin{cases} u_i^1: & r_i & p_{j_i} & p_{k_i} & t_i & (1 \le i \le n_1) \\ u_i^2: & r_i & p_{k_i} & p_{j_i} & & (1 \le i \le n_1) \\ s_i: & q_i & & & & (1 \le i \le n_2) \end{cases}$$

$$\text{Lecturer preferences:} \quad \begin{cases} w_j: & p_j & q_j & & (1 \le j \le n_2) \\ x_j: & r_j & & & (1 \le j \le n_1) \\ y_j: & t_j & & & (1 \le j \le n_1) \end{cases}$$

Figure 3: Preference lists for the constructed instance of SPA-P.

We construct an instance I of SPA-P as follows: let $U^1 \cup U^2 \cup S$ be the set of students, where $U^z = \{u_1^z, u_2^z, \ldots, u_{n_1}^z\}$ $(1 \le z \le 2)$ and $S = \{s_1, s_2, \ldots, s_{n_2}\}$; let $P \cup Q \cup R \cup T$ be the set of projects, where $P = \{p_1, p_2, \ldots, p_{n_2}\}$, $Q = \{q_1, q_2, \ldots, q_{n_2}\}$, $R = \{r_1, r_2, \ldots, r_{n_1}\}$ and $T = \{t_1, t_2, \ldots, t_{n_1}\}$; and let $W \cup X \cup Y$ be the set of lecturers, where $X = \{x_1, x_2, \ldots, x_{n_1}\}$ and $Y = \{y_1, y_2, \ldots, y_{n_1}\}$. Each project and lecturer has capacity 1. The preference lists in I are shown in Figure 3. Clearly the length of each student's list is at most 4, whilst the length of each lecturer's list is at most 2. These lists also indicate the acceptable projects for each student, and the projects offered by each lecturer. We claim that $s^+(I) = 2n_1 + n_2 - \beta_1^-(G)$.

Suppose firstly that G has a maximal matching M, where $|M| = \beta_1^-(G)$. We construct a matching M' in I as follows. For each edge $\{u_i, w_j\}$ in M, if $j = j_i$, add (u_i^1, p_{j_i}) and (u_i^2, r_i) to M'. If $j = k_i$, add (u_i^1, r_i) and (u_i^2, p_{k_i}) to M'. For each $u_i \in U$, if u_i is unassigned in M, add (u_i^1, t_i) and (u_i^2, r_i) to M'. For each $w_j \in W$, if w_j is unassigned in M, add (s_j, q_j) to M'.

No project in $Q \cup R$ can be involved in a blocking pair of M', since each member of $W \cup X$ is full in M'. Hence no student in S can be involved in a blocking pair of M'. Similarly no $u_i^2 \in U^2$ can be involved in a blocking pair of M', since u_i^2 is assigned in M' to either his first or second choice. Also no project in T can be involved in a blocking pair of M', since each member of U^1 is assigned in M'. Now suppose that (u_i^1, p_j) blocks M'. Then $(u_i^1, t_i) \in M'$ and p_j is under-subscribed. Thus no edge of M is incident to u_i or w_j in G. Hence $M \cup \{\{u_i, w_j\}\}$ is a matching in G, contradicting the maximality of M. Thus M' admits no blocking pair.

We next verify that M' is also coalition-free. Clearly no student in S can be involved in a coalition, since any such student who is assigned in M has his first choice. Similarly no student who is assigned in M to a project in R can be in a coalition. As a consequence no student in U who has his second choice can be in a coalition, since each such student prefers only a project in R. Finally, no student in U^1 who has his fourth choice can be in a coalition, since no assigned student prefers a project in T to his project in M. Hence M' is stable. Finally we note that $|M'| = 2|M| + 2(n_1 - |M|) + (n_2 - |M|) = 2n_1 + n_2 - |M|$, and hence $s^+(I) \ge 2n_1 + n_2 - \beta_1^-(G)$.

Conversely, suppose that M' is a stable matching in I such that $|M'| = s^+(I)$. For each $r_j \in R$, it follows that r_j is assigned either u_j^1 or u_j^2, for otherwise (u_j^1, r_j) blocks M', a contradiction. Hence

$$M = \left\{ \{u_i, w_j\} \in E : (u_i^1, p_j) \in M' \ \vee \ (u_i^2, p_j) \in M' \right\}$$

is a matching in G. Suppose that M is not maximal. Then there is some edge $\{u_i, w_j\}$ in G such that no edge of M is incident to u_i or w_j. Thus by construction of M, either (i) $(u_i^1, t_i) \in M'$, or u_i^1 is unassigned in M', or (ii) u_i^2 is unassigned in M'. Also p_j is under-subscribed, and either w_j is under-subscribed or $(s_j, q_j) \in M'$. In Case (i), it follows that (u_i^1, p_j) blocks M', whilst in Case (ii), it follows that (u_i^2, p_j) blocks M'. This contradiction to the stability of M' implies that M is indeed maximal.

For each $\{u_i, w_j\} \in M$, it follows that $(u_i^z, p_j) \in M'$ for some z $(1 \leq z \leq 2)$. Thus $(u_i^{3-z}, r_i) \in M'$. Hence at most $n_1 - |M|$ projects in T are full in M'. Also by construction of M, it follows that $|M|$ projects in P are full in M'. Hence at most $n_2 - |M|$ projects in Q are full in M'. It follows that $|M'| \leq |M| + (n_2 - |M|) + n_1 + (n_1 - |M|) = 2n_1 + n_2 - |M|$ and thus $s^+(I) \leq 2n_1 + n_2 - \beta_1^-(G)$.

Hence $s^+(I) + \beta_1^-(G) = 2n_1 + n_2$. Now $2n_1 = 3n_2$, as G is the subdivision graph of the cubic graph G'. Also $m = 2n_1$, where m is the number of edges in G. Let n be the number of students in I. Then $n = 2n_1 + n_2$.

Let c_0 and δ_0 be the constants given by Theorem 1, such that it is NP-hard to distinguish between the cases $\beta_1^-(G) \leq c_0 m$ and $\beta_1^-(G) > \delta_0 c_0 m$. Hence if $\beta_1^-(G) \leq c_0 m$, then $s^+(I) \geq c_1 n$, whilst if $\beta_1^-(G) > \delta_0 c_0 m$, then $s^+(I) < (1/\delta_1) c_1 n$, where $c_1 = \frac{4-3c_0}{4}$ and $\delta_1 = \frac{4-3c_0}{4-3\delta_0 c_0}$. The result then follows by Theorem 1 and Proposition 3. □

4 Approximation algorithm

The NP-hardness of MAX-SPA-P naturally leads to the question of the approximability of this problem. In this section we present an approximation algorithm for MAX-SPA-P that has a performance guarantee of 2.

Consider the algorithm SPA-P-approx shown in Figure 4, which is an extension of the resident-oriented Gale/Shapley algorithm for the Hospitals/Residents problem [6, Section 1.6.3]. Our algorithm involves a sequence of *apply* and *delete* operations to obtain a stable matching that is at least half the size of optimal. At each iteration of the while loop, some unassigned student s_i with a non-empty preference list applies to the first project p_j on his list. If p_j is full, then s_i is rejected and p_j is deleted from s_i's list. If l_k is full (where l_k offers p_j), and p_j is l_k's worst non-empty project, then s_i is also rejected and p_j is deleted from s_i's list. Otherwise s_i becomes provisionally assigned to p_j. If l_k becomes over-subscribed as a result of this assignment, then l_k rejects an arbitrary student s_r from p_z, and p_z is deleted from s_r's list. Next, if l_k is full (irrespective of whether l_k was over-subscribed earlier in the same loop iteration), then each project p_t that l_k finds less desirable than his worst non-empty project is deleted from the preference list of each student who finds p_t acceptable.

We will show that SPA-P-approx produces a stable matching at least half the size of optimal. Firstly, using the following four lemmas, we prove that the algorithm returns a matching (Lemma 5) that admits no coalition (Lemma 6) and no blocking pair (Lemmas 7 and 8), and therefore the algorithm returns a stable matching.

Lemma 5. SPA-P-approx *terminates with a matching.*

Proof. Clearly the while loop terminates. For, at the beginning of some loop iteration, let s_i be a student who is unassigned and has a non-empty list, and let p_j

```
M = ∅;
while (some student s_i is unassigned and s_i has a non-empty list) {
    p_j = first project on s_i's list;
    l_k = lecturer who offers p_j;
    p_z = l_k's worst project;
    if (l_k is non-empty)
        p_z = l_k's worst non-empty project;
    /* s_i applies to p_j */
    if (p_j is full or (l_k is full and p_j = p_z))
        delete p_j from s_i's list;                          -- (1)
    else {
        M = M ∪ {(s_i, p_j)};
        /* s_i is provisionally assigned to p_j and to l_k */
        if (l_k is over-subscribed) {
            s_r = some student in M(p_z);
            M = M\{(s_r, p_z)};
            delete p_z from s_r's list;                      -- (2)
        }
        if (l_k is full) {
            p_z = l_k's worst non-empty project;
            for (each successor p_t of p_z on l_k's list)
                for (each student s_r who finds p_t acceptable)
                    delete p_t from s_r's list;              -- (3)
        }
    }
}
```

Figure 4: Approximation algorithm SPA-P-approx for MAX-SPA-P.

be the first project on s_i's list. If s_i does not become provisionally assigned to p_j during the same loop iteration, then p_j is removed from s_i's list. If s_i does become provisionally assigned to p_j during this loop iteration then some student s_r may become unassigned; in this case p_j is deleted from s_r's list. Hence eventually, we are guaranteed that each student is either assigned to some project or has an empty list. Let M be the assignment relation upon termination of SPA-P-approx. It is immediate that each student is assigned to at most one project in M, whilst no project or lecturer is over-subscribed in M. □

Lemma 6. SPA-P-approx *returns a matching that is coalition-free.*

Proof. By Lemma 5, let M be the matching output by an execution E of SPA-P-approx. Suppose for a contradiction that M admits a coalition $\{s_{i_0}, s_{i_1}, \ldots, s_{i_{r-1}}\}$ for some $r \geq 2$. In what follows, the concept of a pair (s_i, p_j) being *deleted* refers to the operation of p_j being deleted from s_i's preference list during some iteration of the while loop during E. For each j $(0 \leq j \leq r-1)$, $(s_{i_j}, M(s_{i_{j+1}}))$ must have been deleted during an iteration of the while loop during E. Without loss of generality suppose that the coalition is ordered such that $(s_{i_0}, M(s_{i_1}))$ is the first deletion of the form $(s_{i_j}, M(s_{i_{j+1}}))$ $(0 \leq j \leq r-1)$ to take place during E. Let $p_z = M(s_{i_1})$ and let l_k be the lecturer who offers p_z. We consider the following four cases (in what

follows, all instances of the term *deletion point* i, for $i = 1, 2, 3$, refer to the deletion operation commented by (i) in the pseudocode shown in Figure 4).

Case 1: p_z was deleted from s_{i_0}'s list at deletion point 1, as a result of p_z being full during E. Then s_{i_1} must have applied to p_z after s_{i_0} did, for if not, then s_{i_1} was already assigned to p_z when s_{i_0} applied to p_z. Hence $M(s_{i_2})$ must already have been deleted from s_{i_1}'s list, a contradiction, since (s_{i_0}, p_z) is the first deletion of the form $(s_{i_j}, M(s_{i_{j+1}}))$ $(0 \leq j \leq r-1)$. Therefore p_z must have gone from being full to being under-subscribed during E. This can only happen if l_k became over-subscribed during E, and p_z was l_k's worst non-empty project at that point. Thus when s_{i_1} applied to p_z, it follows that p_z was still l_k's worst non-empty project, and l_k was full. Therefore l_k rejected s_{i_1} from p_z, a contradiction. Hence no coalition exists in this case.

Case 2: p_z was deleted from s_{i_0}'s list at deletion point 1, as a result of l_k being full during E, and p_z was l_k's worst non-empty project. As in Case 1, s_{i_1} must have applied to p_z after s_{i_0} did. Thus when s_{i_1} applied to p_z, l_k must have been full and his worst non-empty project was p_z. Hence s_{i_1} was rejected from p_z, a contradiction. Hence no coalition exists in this case.

Case 3: p_z was deleted from s_{i_0}'s list at deletion point 2, as a result of l_k being over-subscribed during E. Then just before the deletion occurred, p_z was l_k's worst non-empty project. The remainder of this case is similar to Case 2.

Case 4: p_z was deleted from s_{i_0}'s list at deletion point 3, as a result of l_k being full during E. Then p_z is removed from s_{i_1}'s list as well, a contradiction. Hence no coalition exists in this case. □

Lemma 7. *Suppose that some project p_t is deleted from a student s_r's list during an execution of* SPA-P-approx. *Then (s_r, p_t) cannot block a matching output by* SPA-P-approx.

Proof. Let l_k be the lecturer who offers p_t. Let E be an execution of the algorithm during which p_t is deleted from s_r's list. By Lemma 5, let M be the matching output at the termination of E. Suppose for a contradiction that (s_r, p_t) blocks M. We consider the following four cases.

Case 1: p_t was deleted from s_r's list at deletion point 1, as a result of p_t being full during E. Since (s_r, p_t) blocks M, p_t is under-subscribed in M. Hence p_t changed from being full during E to being under-subscribed, which can only occur as a result of lecturer l_k being over-subscribed during E, where p_t was l_k's worst non-empty project at that point. Thus l_k is full in M, and l_k's worst non-empty project is either p_t or better. Hence (s_r, p_t) does not block M in this case.

Case 2: p_t was deleted from s_r's list at deletion point 1, as a result of l_k being full during E, and p_t was l_k's worst non-empty project. Clearly on termination of E, l_k is full, and l_k's worst non-empty project is p_t or better. Hence (s_r, p_t) does not block M in this case.

Case 3: p_t was deleted from s_r's list at deletion point 2, as a result of l_k being over-subscribed during E. Then just before the deletion occurred, p_t was l_k's worst non-empty project. Now l_k is full in M, and l_k's worst non-empty project is either p_t or better. Hence (s_r, p_t) does not block M in this case.

Case 4: p_t was deleted from s_r's list at deletion point 3, as a result of l_k being full during E. Then l_k is full in M, and l_k prefers his worst non-empty project to p_t. Hence (s_r, p_t) does not block M in this case. □

Lemma 8. SPA-P-approx *returns a stable matching.*

Proof. Let E be an execution of the algorithm, and by Lemma 5, let M be the matching output upon termination of E. Suppose that (s_i, p_j) blocks M. By Lemma 7, p_j is not deleted from s_i's list during E. Hence s_i's list is non-empty upon termination of E. If s_i is unassigned in M then the while loop would not have terminated, a contradiction. Hence s_i is assigned in M and prefers p_j to $p_r = M(s_i)$. But when s_i applied to p_r, p_r was the first project on s_i's list, a contradiction. Hence, and by Lemma 6, M is stable. □

It follows by Lemma 8 that SPA-P-approx is an approximation algorithm for MAX-SPA-P. Moreover, using a suitable choice of data structures, the algorithm may be implemented to run in time linear in the length of the given preference lists. The following is therefore immediate.

Corollary 9. *Every instance I of* SPA-P *admits at least one stable matching, and such a matching may be found in $O(\lambda)$ time, where λ is the total length of the preference lists in I.*

The next result shows that SPA-P-approx has a performance guarantee of 2.

Lemma 10. *Let I be an instance of* SPA-P. *Then $|M| \leq 2|M'|$ for any stable matchings M, M' in I.*

Proof. Suppose for a contradiction that $|M'| < |M|/2$. Let X (respectively Y) be those students who are assigned in M but not M' (respectively M' but not M), and let Z be those students who are assigned in both M and M'. Then

$$|X| = |M| - |Z| > 2|M'| - |Z| = 2|Y| + |Z| \geq |M'|. \tag{1}$$

Now suppose that the students in X are collectively assigned in M to projects $P' = \{p_1, \ldots, p_s\}$ offered by lecturers $l_1, \ldots l_t$. Suppose that $P'_1, \ldots, P'_t$ is a partition of P' such that lecturer l_k $(1 \leq k \leq t)$ offers the projects in P'_k. Similarly let $S_1, \ldots, S_t$ be a partition of X such that each student in S_k is assigned in M to a project in P'_k $(1 \leq k \leq t)$.

Now let k be given $(1 \leq k \leq t)$ and let p_j be any project in P'_k. Then there is some student $s_i \in S_k$ who is assigned to p_j in M but unassigned in M'. Hence in M', either (i) p_j is full, or (ii) l_k is full (or both), for otherwise (s_i, p_j) blocks M'. It follows that, in M', either (a) all projects in P'_k are full, or (b) l_k is full (or both). Hence

$$|M'| \geq \sum_{k=1}^{t} \min \left(d_k, \sum_{p_j \in P'_k} c_j \right). \tag{2}$$

Since no project or lecturer is over-subscribed in M, it follows that, for each k $(1 \leq k \leq t)$, $\sum_{p_j \in P'_k} c_j \geq |S_k|$ and $d_k \geq |S_k|$. Hence (2) implies that $|M'| \geq \sum_{k=1}^{t} |S_k| = |X|$, which is a contradiction to (1). Thus $|M'| \geq |M|/2$ as required. □

The above lemmas lead to the following conclusion.

Student preferences

$s_{2i-1}: \quad p_{2i-1} \quad p_{2i} \quad (1 \leq i \leq n)$

$s_{2i}: \quad p_{2i-1} \quad (1 \leq i \leq n)$

Lecturer preferences

$l_k: \quad p_{2k-1} \quad p_{2k} \quad (1 \leq k \leq n)$

$c_j = 1 \ (1 \leq j \leq 2n)$

$d_k = 2 \ (1 \leq k \leq n)$

Figure 5: An instance I_3 of SPA-P.

Theorem 11. SPA-P-approx *is an approximation algorithm for* MAX-SPA-P *with a performance guarantee of 2.*

To demonstrate that the analysis given in the proof of Lemma 10 is tight, it is straightforward to construct an instance of SPA-P such that the algorithm SPA-P-approx could produce a stable matching that is half the size of optimal. For, consider the instance of SPA-P shown in Figure 5, where $\mathcal{S} = \{s_1, \ldots, s_{2n}\}$, $\mathcal{P} = \{p_1, \ldots, p_{2n}\}$ and $\mathcal{L} = \{l_1, \ldots, l_n\}$. The matching $M = \{(s_{2i-1}, p_{2i}), (s_{2i}, p_{2i-1}) : 1 \leq i \leq n\}$ is the unique maximum stable matching, of size $2n$. On the other hand, during an execution of SPA-P-approx, if the students apply to projects in increasing indicial order, we obtain the stable matching $M' = \{(s_{2i-1}, p_{2i-1}) : 1 \leq i \leq n\}$, of size n.

5 Concluding remarks

In this paper we have considered a model for the Student-Project Allocation problem (SPA) in which both students and lecturers have preferences over projects. As noted in Section 1, a SPA model in which lecturers have preferences over students has also been studied [1, 2]. It remains to investigate algorithmic issues for a more general preference model for the lecturers, involving preferences over (student,project) pairs. Some detailed initial observations regarding this case are made in [2].

For the SPA-P model, involving lecturer preferences over projects, this paper showed that the problem of finding a maximum stable matching is NP-hard, though admits an approximation algorithm, SPA-P-approx, with a performance guarantee of 2. In practice, SPA-P-approx is likely to construct a stable matching whose size is closer to optimal than a factor of 1/2, nevertheless the question remains as to whether there exists an approximation algorithm for MAX-SPA-P that has a performance guarantee less than 2.

Acknowledgement

We would like to thank Rob Irving and an anonymous referee for helpful comments on previous drafts of this paper.

References

[1] D.J. Abraham, R.W. Irving, and D.F. Manlove. The Student-Project Allocation Problem. In *Proceedings of ISAAC 2003: the 14th Annual International*

Symposium on Algorithms and Computation, volume 2906 of *Lecture Notes in Computer Science*, pages 474–484. Springer-Verlag, 2003.

[2] D.J. Abraham, R.W. Irving, and D.F. Manlove. Two algorithms for the Student-Project allocation problem. To appear in *Journal of Discrete Algorithms*, 2006.

[3] A.A. Anwar and A.S. Bahaj. Student project allocation using integer programming. *IEEE Transactions on Education*, 46(3):359–367, 2003.

[4] J. Dye. A constraint logic programming approach to the stable marriage problem and its application to student-project allocation. BSc Honours project report, University of York, Department of Computer Science, 2001.

[5] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

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

[7] M. Halldórsson, R.W. Irving, K. Iwama, D.F. Manlove, S. Miyazaki, Y. Morita, and S. Scott. Approximability results for stable marriage problems with ties. *Theoretical Computer Science*, 306(1-3):431–447, 2003.

[8] 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.

[9] D. Kazakov. Co-ordination of student-project allocation. Manuscript, University of York, Department of Computer Science, 2002.

[10] L.G. Proll. A simple method of assigning projects to students. *Operational Research Quarterly*, 23(2):195–201, 1972.

[11] 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.

[12] 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.

[13] M. Thorn. A constraint programming approach to the student-project allocation problem. BSc Honours project report, University of York, Department of Computer Science, 2003.

[14] V.V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.