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, and the problem of finding a maximum cardinality stable matching is NP-hard, though approximable within a factor of 2.

Keywords: Matching problem; Stable matching; NP-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). 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, 10, 14], the University of Southampton [3, 8] and elsewhere [13].

[*]Supported by Engineering and Physical Sciences Research Council grant GR/R84597/01.
[†]Supported by Royal Society of Edinburgh/Scottish Executive Personal Research Fellowship.

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 [12]. 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 [12] that, when preference lists exist on two sides of a 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 also described. This algorithm finds 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 the 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/her 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. 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, even in the special case that each project and lecturer can accommodate only one student. 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.

Student preferences	Lecturer preferences
$s_1: \ p_1 \ \ p_2$	$l_1: \ p_1$
$s_2: \ p_1$	$l_2: \ p_2$

Each project and lecturer has capacity 1

Figure 1: An instance I_1 of SPA-P.

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 [11, 13, 3, 8] (so stability is not relevant in these contexts) or involve lecturer preferences over students [4, 10, 1, 14, 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 an associated *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 any $r \in \mathcal{S} \cup \mathcal{P} \cup \mathcal{L}$, 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* in M 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 $l_k \in \mathcal{L}$ (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:

1. $p_j \in A_i$ (i.e. s_i finds p_j acceptable).
2. Either s_i is unassigned in M 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.

A matching is *stable* if it admits no blocking pair. 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).

It turns out that, with respect to this definition, for a given instance of SPA-P, stable matchings could 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 NP-hardness of finding a maximum stable matching

Denote by MAX-SPA-P the problem of finding a maximum stable matching, given an instance of SPA-P. In this section we show that MAX-SPA-P is NP-hard. This follows immediately from the NP-completeness of ALL-SPA-P, which is the problem of deciding, given an instance of SPA-P, whether a stable matching exists in which all students are assigned.

In order to show that ALL-SPA-P is NP-complete, we use 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. Define MIN-MM (respectively EXACT-MM) to be the problem of deciding, given a graph G and integer K, whether G admits a maximal matching of size at most (respectively exactly) K. MIN-MM is NP-complete, even for subdivision graphs [9] (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). We now show that EXACT-MM is NP-complete for the same class of graphs.

Lemma 1. EXACT-MM *is NP-complete, even for subdivision graphs.*

Proof. Clearly EXACT-MM belongs to NP. To show NP-hardness, we reduce from MIN-MM restricted to subdivision graphs, which is NP-complete as mentioned above. Let G (a subdivision graph of some graph G') and K (a positive integer) be an instance of MIN-MM. Without loss of generality we may assume that $K \leq \beta(G)$, where $\beta(G)$ denotes the size of a maximum matching of G. Suppose that G admits a maximal matching M, where $|M| = k \leq K$. If $k = K$, we are done. Otherwise suppose that $k < K$. We note that maximal matchings satisfy the interpolation property [7] (i.e. G has a maximal matching of size j, for $k \leq j \leq \beta(G)$) and hence G has a maximal matching of size K. The converse is clear. □

We now use the NP-completeness of EXACT-MM to establish the same result for ALL-SPA-P.

Theorem 2. ALL-SPA-P *is NP-complete.*

Proof. Clearly ALL-SPA-P belongs to NP. To show NP-hardness, we transform from EXACT-MM restricted to subdivision graphs, which is NP-complete by Lemma 1. Hence let G (a subdivision graph of some graph G') and K (a positive integer) be an instance of EXACT-MM. Then G is a bipartite graph, so that $G = (U, W, E)$, where without loss of generality all vertices in U have degree 2. Suppose that $n_1 = |U|$ and $n_2 = |W|$. Again, without loss of generality assume that $K \leq \min\{n_1, n_2\}$. 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 $j_i < k_i$.

We construct an instance I of ALL-SPA-P as follows: let $U \cup U' \cup V$ be the set of students, where $U' = \{u'_1, u'_2, \ldots, u'_{n_1}\}$ and $V = \{v_1, v_2, \ldots, v_{n_2-K}\}$; let $P \cup Q \cup R \cup S$ 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 $S = \{s_1, s_2, \ldots, s_{n_1-K}\}$; and let $W \cup X \cup Y$ be the set of lecturers, where $X = \{x_1, x_2, \ldots, x_{n_1}\}$, and $Y = \{y_1, y_2, \ldots, y_{n_1-K}\}$. Each project and lecturer has capacity 1. The preference lists in I are shown in Figure 2. These preference lists also indicate the acceptable projects for each student, and the projects offered by each lecturer. In a given preference list, projects within square brackets are listed in arbitrary strict order at the point where the symbol appears. We claim that G has a maximal matching of size K if and only if I admits a stable matching in which all students are assigned.

For, suppose that G has a maximal matching M, where $|M| = K$. We construct a matching M' in I as follows. For each edge $\{u_i, w_j\}$ in M, if $j = j_i$, then we add (u_i, p_{j_i}) and (u'_i, r_i) to M'. If $j = k_i$, then we add (u'_i, p_{k_i}) and (u_i, r_i) to M'. There remain $n_2 - K$ lecturers in W who are under-subscribed in M'. Denote these lecturers by w_{t_j} $(1 \leq j \leq n_2 - K)$. Add (v_j, q_{t_j}) to M' $(1 \leq j \leq n_2 - K)$.

$$\text{Student preferences:} \quad \begin{cases} u_i : & r_i \;\; p_{j_i} \;\; p_{k_i} \;\; [S] & (1 \le i \le n_1) \\ u'_i : & r_i \;\; p_{k_i} & (1 \le i \le n_1) \\ v_i : & [Q] & (1 \le i \le n_2 - K) \end{cases}$$

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

Figure 2: Preference lists for the constructed instance of ALL-SPA-P.

Similarly there remain $2(n_1 - K)$ students in $U \cup U'$ who are unassigned in M'. Denote these students by u_{z_i}, u'_{z_i} $(1 \le i \le n_1 - K)$. Add (u_{z_i}, s_i) and (u'_{z_i}, r_{z_i}) to M' $(1 \le i \le n_1 - K)$. Clearly M' is a matching in I in which all students are assigned.

No project in $Q \cup R \cup S$ can be involved in a blocking pair of M', since each member of $W \cup R \cup S$ is full in M'. Hence no student in $U' \cup V$ can be involved in a blocking pair of M', since every student is assigned in M'. Finally, no pair $(u_i, p_j) \notin M'$ blocks M', where $u_i \in U$ and $p_j \in P$. For if this occurs, then $(u_i, s_l) \in M'$ for some $s_l \in S$, 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' is stable.

Conversely, suppose that M' is a stable matching in I in which all students are assigned. For each $r_j \in R$, it follows that r_j is assigned either u_j or u'_j, for otherwise (u_j, r_j) blocks M', a contradiction. Hence

$$M = \{\{u_i, w_j\} \in E : (u_i, p_j) \in M' \;\vee\; (u'_i, p_j) \in M'\}$$

is a matching in G. Now each student in V is assigned in M' to a project in Q, so $n_2 - K$ projects in Q are full in M'. Hence at most K projects in P are full in M', since each lecturer in W has capacity 1. Now in M', at most $n_1 - K$ students in U are assigned to projects in S. As already observed, exactly n_1 students in $U \cup U'$ are assigned in M' to projects in R. Hence at least K students in $U \cup U'$ are assigned in M' to projects in P, so that $|M| = K$.

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 $(u'_i, r_i) \in M'$, so that $(u_i, s_l) \in M'$ for some $s_l \in S$. Also either w_j is under-subscribed, or $(v_k, q_j) \in M'$ for some $v_k \in V$. Hence (u_i, p_j) blocks M', for p_j is under-subscribed. This contradiction to the stability of M' implies that M is indeed maximal. □

The following corollary is an immediate consequence of Theorem 2.

Corollary 3. MAX-SPA-P *is NP-hard, even if each project and lecturer has capacity 1.*

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.

```
$M = \emptyset$;
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 $p_j$;
    /* $s_i$ applies to $p_j$ */
    if ($p_j$ is full)
        delete $p_j$ from $s_i$'s list;
    else {
        $M = M \cup \{(s_i, p_j)\}$;
        /* $s_i$ is provisionally assigned to $p_j$ and to $l_k$ */
        if ($l_k$ is over-subscribed) {
            $p_z$ = $l_k$'s worst non-empty project;
            $s_r$ = some student in $M(p_z)$;
            $M = M \backslash \{(s_r, p_z)\}$;
            delete $p_z$ from $s_r$'s list;
        }
        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;
        }
    }
}
```

Figure 3: Approximation algorithm SPA-P-approx for MAX-SPA-P.

Consider the algorithm SPA-P-approx, as shown in Figure 3, 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* operations, in which an unassigned student s_i with a non-empty list applies to the first project p_j on his list. If p_j is full, then s_i is rejected, and must apply to the next project in his list, if such a project exists. If p_j is under-subscribed, then s_i is provisionally assigned to p_j. If, as a result of this assignment, lecturer l_k becomes over-subscribed (where l_k offers p_j), then l_k rejects an arbitrary student s_r from his worst non-empty project p_z, and p_z is deleted from s_r's list. If l_k is full (irrespective of whether l_k was over-subscribed earlier in the same loop iteration), we set p_z to be l_k's worst non-empty project. For each successor p_t of p_z on l_k's list, we delete p_t from the preference list of every student who finds p_t acceptable.

We will show that algorithm SPA-P-approx produces a stable matching at least half the size of optimal. Firstly, using the following three lemmas, we prove that the algorithm returns a stable matching.

Lemma 4. SPA-P-approx *returns a matching.*

Proof. Clearly the while loop terminates. For, at the beginning of some loop iteration, let s_i be a student who is free and has a non-empty list, and let p_j 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. 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 5. *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 E be an execution of the algorithm during which p_t is deleted from s_r's list. By Lemma 4, let M be the matching output at the termination of E. Suppose for a contradiction that (s_r, p_t) blocks M. We consider three cases.

Case 1: p_t was deleted from s_r's list 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 some 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 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 3: p_t was deleted from s_r's list as a result of l_k being full during E. Then l_k is full in M, and l_k prefers his/her worst non-empty project to p_t. Hence (s_r, p_t) does not block M in this case. □

Lemma 6. SPA-P-approx *returns a stable matching.*

Proof. Let E be an execution of the algorithm, and by Lemma 4, let M be the matching output upon termination of E. Suppose that (s_i, p_j) blocks M. By Lemma 5, 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 M is stable. □

It follows by Lemma 6 that SPA-P-approx is an approximation algorithm for MAX-SPA-P. Moreover a further consequence of this lemma is that every instance of SPA-P admits at least one stable matching. The next result shows that SPA-P-approx has a performance guarantee of 2.

Theorem 7. SPA-P-approx *is an approximation algorithm for* MAX-SPA-P *with a performance guarantee of 2.*

Proof. Let I be an instance of SPA-P and let M be a stable matching of maximum size in I. By Lemma 6, let M' be a stable matching output by SPA-P-approx as applied to I, and 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

$$|X| = |M| - |Z| > 2|M'| - |Z| = 2|Y| + |Z| \geq |M'|. \qquad (1)$$

Student preferences			Lecturer preferences		
$s_{2i-1}:$	$p_{2i-1}\ p_{2i}$	$(1 \leq i \leq n)$	$l_j:$	$p_{2j-1}\ p_{2j}$	$(1 \leq j \leq n)$
$s_{2i}:$	p_{2i-1}	$(1 \leq i \leq n)$			

Each project has capacity 1
Each lecturer has capacity 2

Figure 4: An instance I_2 of SPA-P.

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 Inequality 2 implies that $|M'| \geq \sum_{k=1}^{t} |S_k| = |X|$, which is a contradiction to Inequality 1. Thus $|M'| \geq |M|/2$ as required. □

To demonstrate that the analysis given in the proof of Theorem 7 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 4, 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 $\frac{1}{2}$, 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.

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. Submitted to *Journal of Discrete Algorithms*, 2004.

[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] F. Harary. Maximum versus minimum invariants for graphs. *Journal of Graph Theory*, 7:275–284, 1983.

[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] J.D. Horton and K. Kilakos. Minimum edge dominating sets. *SIAM Journal on Discrete Mathematics*, 6:375–387, 1993.

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

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

[12] 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.

[13] 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.

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