Profile-based optimal matchings in the Student/Project Allocation problem

Augustine Kwanashie[1], Robert W. Irving[1],
David F. Manlove[1,⋆] and Colin T.S. Sng[2,⋆⋆]

[1]School of Computing Science, University of Glasgow, UK,
[2]eBay, Inc., Austin, Texas, USA.

Abstract. In the *Student/Project Allocation problem* (SPA) we seek to assign students to individual or group projects offered by lecturers. Students provide a list of projects they find acceptable in order of preference. Each student can be assigned to at most one project and there are constraints on the maximum number of students that can be assigned to each project and lecturer. We seek matchings of students to projects that are optimal with respect to *profile*, which is a vector whose rth component indicates how many students have their rth-choice project. We present an efficient algorithm for finding a *greedy maximum matching* in the SPA context – this is a maximum matching whose profile is lexicographically maximum. We then show how to adapt this algorithm to find a *generous maximum matching* – this is a matching whose reverse profile is lexicographically minimum. Our algorithms involve finding optimal flows in networks. We demonstrate how this approach can allow for additional constraints, such as lecturer lower quotas, to be handled flexibly.

1 Introduction

In most academic programmes students are usually required to take up individual or group projects offered by lecturers. Students are required to rank a subset of the projects they find acceptable in order of preference. Each project is offered by a unique lecturer who may also be allowed to rank the projects she offers or the students who are interested in taking her projects in order of preference. Each student can be assigned to at most one project and there are usually constraints on the maximum number of students that can be assigned to each project and lecturer. The problem then is to assign students to projects in a manner that satisfies these capacity constraints while taking into account the preferences of the students and lecturers involved. This problem has been described in the literature as the *Student-Project Allocation problem* (SPA). In some cases, lecturer *lower quotas*, indicating the minimum number of students to be assigned to each lecturer, may also be specified.

⋆ Supported by Engineering and Physical Sciences Research Council grant EP/K010042/1.

⋆⋆ Work done while at the School of Computing Science, University of Glasgow.

Although described in an academic context, applications of SPA need not be limited to assigning students to projects but may extend to other scenarios, such as the assignment of employees to posts in a company where available posts are offered by various departments. It is widely accepted that matching problems (like SPA) are best solved by centralised matching schemes where agents submit their preferences and a central authority computes an optimal matching that satisfies all the specified criteria [5]. Moreover the potentially large number of students and projects involved in these schemes motivates the need to discover efficient algorithms for finding optimal matchings.

In SPA, students are always required to provide preference lists over projects. However, variants of the problem may be defined depending on the presence and nature of lecturer preference lists. Some variants of SPA require both students and lecturers to provide preference lists. These variants include: (i) the *Student/Project Allocation problem with lecturer preferences over Students* (SPA-S) [2] which requires each lecturer to rank the students who find at least one of her offered projects acceptable, in order of preference, (ii) the *Student/Project Allocation problem with lecturer preferences over Projects* (SPA-P) [14,11] which involves lecturers ranking the projects they offer in order of preference and (iii) the *Student/Project Allocation problem with lecturer preferences over Student-Project pairs* (SPA-(S,P)) [2,3] where lecturers rank student-project pairs in order of preference. These variants of SPA have been studied in the context of the well-known *stability* solution criterion for matching problems [5]. The general stability objective is to produce a matching M in which no student-project pair that are not currently matched in M can simultaneously improve by being paired together (thus in the process potentially abandoning their partners in M). A full description of the results relating to these SPA variants can be found in [13].

1.1 One-sided preferences and profile-based optimality

In many practical SPA applications it is considered appropriate to allow only students to submit preferences over projects. When preferences are specified by only one set of agents in a two-sided matching problem, the notion of stability becomes irrelevant. This motivates the need to adopt alternative solution criteria when lecturer preferences are not allowed. In this subsection we mention some of these solution criteria and briefly present results relating to them. These criteria consider the size of the matchings produced as well as the satisfaction of the students involved.

When the preference lists of the lecturers are ignored, the SPA problem becomes a two-sided matching problem with one-sided preferences. Various optimality criteria for such problems have been studied in the literature [13]. Some of these criteria depend on the *profile* or the *cost* of a matching. In the SPA context, the profile of a matching is a vector whose rth component indicates the number of students obtaining their rth-choice project in the matching. The cost of a matching (w.r.t. the students) is the sum of the ranks of the assigned projects in the students' preference lists (that is, the sum of rx_r taken over all components r of the profile, where x_r is the rth component value). A *minimum*

cost maximum matching is a maximum cardinality matching with minimum cost. A *rank-maximal matching* is a matching that has lexicographically maximum profile [10,8]. That is the maximum number of students are assigned to their first-choice project and subject to this, the maximum number of students are assigned to their second choice project and so on. However a rank maximal matching need not be a maximum matching in the given instance (see, e.g., [13, p.43]). Since it is usually important to match as many students as possible, we may first optimise the size of the matching before considering student satisfaction. Thus we define a *greedy maximum matching* [9,15,6] as a maximum matching which has lexicographically maximum profile. The intuition behind both rank-maximal and greedy maximum matchings is to maximize the number of students matched with higher ranked projects. This may lead to some students being matched to projects that are relatively low on their preference lists. An alternative approach is to find a *generous maximum matching* which is a maximum matching in which the minimum number of students are matched to their Rth-choice project (where R is the maximum length of any students' preference list) and subject to this, the minimum number of students are matched to their $(R-1)$th-choice project and so on. Greedy and generous maximum matchings have been used to assign students to projects in the School of Computing Science, and students to elective courses in the School of Medicine, both at the University of Glasgow, since 2007.

A special case of SPA, where each project is offered by a unique lecturer with an infinite upper quota and zero lower quota, can be modelled as the *Capacitated House Allocation problem* (CHA). This is a variant of the well-studied *House Allocation problem* (HA) [7,19] which involves the allocation of a set of indivisible goods (which we call houses) to a set of applicants. In CHA, each applicant is required to rank a subset of the houses in order of preference with the houses having no preference over applicants. The applicants play the role of students and the houses play the role of projects and lecturers. As in the case of SPA, we seek to find a many-to-one matching comprising applicant-house pairs. Efficient algorithms for finding profile-based optimal matchings in CHA have been studied in the literature [9,15,17,6]. The most efficient of these is the $O(R^*m\sqrt{n})$ algorithm for finding rank-maximal, greedy maximum and generous maximum matchings in CHA problems due to Huang et al [6] where R^* is the maximum rank of any applicant in the matching, m is the sum of all the preference list lengths and n is the total number of applicants and houses. These models however fail to address the issue of load balancing among lecturers. In order to keep the assignment of students fair each lecturer will typically have a minimum (lower quota) and maximum (capacity/upper quota) number of students they are expected to supervise. These numbers may vary for different lecturers according to other administrative and academic commitments. Finding efficient algorithms for profile-based optimal matchings when considering these lecturer upper and lower quotas is the main motivation of this paper.

The CHA algorithms mentioned above are based on modelling the problem in terms of a bipartite graph with the aim of finding a matching in the graph which

satisfies the stated criteria. However a more flexible approach would be to model the problem as a network with the aim of finding a flow that can be converted to a matching which satisfies the stated criteria. SPA has also been investigated in the network flow context [1,18] where a *minimum cost maximum flow algorithm* is used to find a minimum cost maximum matching and other profile-based optimal matchings. The model presented in [18] allows for lower quotas on lecturers and projects as well as alternative lecturers to supervise each project. By an appropriate assignment of edge weights in the network it is shown that a minimum cost maximum flow algorithm (due to Orlin [16]) can find rank maximal, generous maximum and greedy maximum matchings in a SPA instance. This takes $O(m \log n(m + n \log n))$ time in the worst case, where m and n are the number of vertices and edges in the network respectively. In the SPA context this takes $O(m_2^2 \log(n_1 + n_2) + m_2(n_1 + n_2) \log^2(n_1 + n_2))$ time, where n_1 is the number of students, n_2 is the number of projects and m_2 is the sum of all the students' preference list lengths. However this approach involves assigning exponentially large edge weights (see, e.g., [13, p.405]), which may be computationally infeasible for larger problem instances due to floating point inaccuracies in dealing with such high numbers. For example given a large SPA instance involving say, $n_1 = 100$ students each ranking $R = 10$ projects in order of preference, edge weights could potentially be of the order $n_1^R = 100^{10} = 10^{20}$ (and arithmetic involving such weights could easily require more than the 15-17 significant figures available in a 64-bit double-precision floating representation). Since the flow algorithms involve comparing these edge weights, floating point precision errors could easily cause them to fail in practice. Moreover using the standard assumption that arithmetic on numbers of magnitude $O(n_1)$ takes constant time, arithmetic on edge weights of magnitude $O(n_1^R)$ would add an additional factor of $O(R)$ onto the running time of Orlin's algorithm.

1.2 Our contribution

In this paper we present efficient algorithms for finding optimal matchings to SPA problems based on the profile-based greedy maximum and generous maximum optimality criteria. Our model allows for lecturer upper and lower quotas and finds these profile-based optimal matchings without the need for exponentially-large edge weights.

We model SPA as a network flow problem and describe a modified augmenting path algorithm for finding a maximum flow which can then be transformed to an optimal SPA matching. This approach introduces greater flexibility by allowing side constraints like lecturer lower quotas to be added to the model. Our algorithms run in $O(n_1^2 R m_2)$ time. The elimination of large edge weights comes at the expense of a slightly slower running time than that of Orlin's algorithm in the worst case (i.e. slower by a factor of $O(\max\{\frac{n_1}{logn_1}, \frac{n_2}{logn_2}\})$ in all cases. See [12] for further details).

The remainder of this paper is organised as follows. In Section 2 we formally define the model. In Section 3 we describe an efficient algorithm for finding a greedy maximum matching given a SPA instance. In Section 4 we show how

this algorithm can be modified in order to find a generous maximum matching. Finally in Section 5 we explain how the approach can be extended to allow lecturer lower quotas. All proofs for this paper can be found in [12].

2 Preliminary definitions

An instance I of the SPA problem consists of a set $\mathcal{S}$ of students, a set $\mathcal{P}$ of projects and a set $\mathcal{L}$ of lecturers. Each student s_i ranks a set $A_i \subseteq \mathcal{P}$ of projects that she considers acceptable in order of preference. This *preference list* of projects may contain ties. Each project $p_j \in \mathcal{P}$ has an upper quota c_j indicating the maximum number of students that can be assigned to it. Each lecturer $l_k \in \mathcal{L}$ offers a set of projects $P_k \subseteq \mathcal{P}$ and has an upper quota d_k^+ indicating the maximum number of students that can be assigned to l_k. Unless explicitly mentioned, we assume that all lecturer lower quotas are equal to 0. The sets $\{P_1, \ldots, P_k\}$ partition $\mathcal{P}$. If project $p_j \in P_k$, then we denote $l_k = l(p_j)$.

An *assignment* M in I is a subset of $\mathcal{S} \times \mathcal{P}$ such that:

1. Student-project pair $(s_i, p_j) \in M$ implies $p_j \in A_i$.
2. For each student $s_i \in \mathcal{S}$, $|\{(s_i, p_j) \in M : p_j \in A_i\}| \leq 1$.

If $(s_i, p_j) \in M$ we denote $M(s_i) = p_j$. For a project p_j, $M(p_j)$ is the set of students assigned to p_j in M. Also if $(s_i, p_j) \in M$ and $p_j \in P_k$ we say student s_i is assigned to project p_j and to lecturer l_k in M. We denote the set of students assigned to a lecturer l_k as $M(l_k)$. A *matching* in this problem is an assignment M that satisfies the capacity constraints of the projects and lecturers. That is, $|M(p_j)| \leq c_j$ for all projects $p_j \in \mathcal{P}$ and $|M(l_k)| \leq d_k^+$ for all lecturers $l_k \in \mathcal{L}$.

Given a student s_i and a project $p_j \in A_i$, we define $rank(s_i, p_j)$ as 1 + the number of projects that s_i prefers to p_j. Let R be the maximum rank of a project in any student's preference list. We define the *profile* $\rho(M)$ of a matching M in I as an R-tuple $(x_1, x_2, ..., x_R)$ where for each r $(1 \leq r \leq R)$, x_r is the number of students s_i assigned in M to a project p_j such that $rank(s_i, p_j) = r$. Let $\alpha = (x_1, x_2, ..., x_R)$ and $\sigma = (y_1, y_2, ..., y_R)$ be any two profiles. We define the *empty profile* $O_R = (o_1, o_2, ..., o_R)$ where $o_r = 0$ for all r $(1 \leq r \leq R)$. We also define the *negative infinity profile* $B_R^- = (b_1, b_2, ..., b_R)$ where $b_r = -\infty$ $(1 \leq r \leq R)$ and the *positive infinity profile* $B_R^+ = (b_1, b_2, ..., b_R)$ where $b_r = +\infty$ $(1 \leq r \leq R)$. We define the sum of two profiles α and σ as $\alpha + \sigma = (x_1 + y_1, x_2 + y_2, ..., x_R + y_R)$. Given any q $(1 \leq q \leq R)$, we define $\alpha + q = (x_1, ..., x_{q-1}, x_q + 1, x_{q+1}, ..., x_R)$. We define $\alpha - q$ in a similar way.

We define the total order $\succ_L$ on profiles as follows. We say α *left dominates* σ, denoted by $\alpha \succ_L \sigma$ if there exists some r $(1 \leq r \leq R)$ such that $x_{r'} = y_{r'}$ for $1 \leq r' < r$ and $x_r > y_r$. We define *weak left domination* as follows. We say $\alpha \succeq_L \sigma$ if $\alpha = \sigma$ or $\alpha \succ_L \sigma$. We may also define an alternative total order $\prec_R$ on profiles as follows. We say α *right dominates* σ $(\alpha \prec_R \sigma)$ if there exists some r $(1 \leq r \leq R)$ such that $x_{r'} = y_{r'}$ for $r < r' \leq R$ and $x_r < y_r$. We also define *weak right domination* as follows. We say $\alpha \preceq_R \sigma$ if $\alpha = \sigma$ or $\alpha \prec_R \sigma$.

The SPA problem can be modelled as a network flow problem. Given a SPA instance I, we construct a flow network $N(I) = \langle G, c \rangle$ where $G = (V, E)$ is a

directed graph and c is a non-negative capacity function $c : E \to \mathbb{R}^+$ defining the maximum flow allowed through each edge in E. The network consists of a single source vertex v_s and sink vertex v_t and is constructed as follows. Let $V = \{v_s, v_t\} \cup \mathcal{S} \cup \mathcal{P} \cup \mathcal{L}$ and $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where $E_1 = \{(v_s, s_i) : s_i \in \mathcal{S}\}$, $E_2 = \{(s_i, p_j) : s_i \in \mathcal{S}, p_j \in A_i\}$, $E_3 = \{(p_j, l_k) : p_j \in \mathcal{P}, l_k = l(p_j)\}$ and $E_4 = \{(l_k, v_t) : l_k \in \mathcal{L}\}$. We set the capacities as follows: $c(v_s, s_i) = 1$ for all $(v_s, s_i) \in E_1$, $c(s_i, p_j) = 1$ for all $(s_i, p_j) \in E_2$, $c(p_j, l_k) = c_j$ for all $(p_j, l_k) \in E_3$ and $c(l_k, v_t) = d_k^+$ for all $(l_k, v_t) \in E_4$.

We call a path P' from v_s to some project p_j a *partial augmenting path* if P' can be extended by adding the edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ to form an augmenting path with respect to flow f. Given a partial augmenting path P' from v_s to p_j, we define the *profile* of P', denoted $\rho(P')$, as follows:

$$\begin{aligned}\rho(P') = O_R + \textstyle\sum\{rank(s_i, p_j) : (s_i, p_j) \in P' \wedge f(s_i, p_j) = 0\} \\ - \textstyle\sum\{rank(s_i, p_j) : (p_j, s_i) \in P' \wedge f(s_i, p_j) = 1\}\end{aligned}$$

where additions are done with respect to the $+$ and $-$ operations on profiles. Unlike the profile of a matching, the profile of an augmenting path may contain negative values. Also if P' can be extended to a full augmenting path P with respect to flow f by adding the edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ where v_s and p_j are the endpoints of P', then we define the profile of P, denoted by $\rho(P)$, to be $\rho(P) = \rho(P')$. Multiple partial augmenting paths may exist from v_s to p_j, thus we define the *maximum profile of a partial augmenting path* from v_s to p_j with respect to $\succ_L$, denoted $\Phi(p_j)$, as follows:

$$\Phi(p_j) = \max_{\succ_L}\{\rho(P') : P' \text{ is a partial augmenting path from } v_s \text{ to } p_j\}.$$

An augmenting path P is called a *maximum profile augmenting path* if $\rho(P) = \max_{\succ_L}\{\Phi(p_j) : p_j \in \mathcal{P}\}$. Let f be an integral flow in N. We define the matching $M(f)$ in I induced by f as follows: $M(f) = \{(s_i, p_j) : f(s_i, p_j) = 1\}$. Clearly by construction of N, $M(f)$ is a matching in I, such that $|M(f)| = |f|$. If f is a flow and P is an augmenting path with respect to f then $\rho(M') = \rho(M) + \rho(P)$ where $M = M(f), M' = M(f')$ and f' is the flow obtained by augmenting f along P. Also given a matching M in I, we define a flow $f(M)$ in N corresponding to M as follows:

$\forall\ (v_s, s_i) \in E_1$, $f(v_s, s_i) = 1$ if s_i is matched in M and $f(v_s, s_i) = 0$ otherwise.
$\forall\ (s_i, p_j) \in E_2$, $f(s_i, p_j) = 1$ if $(s_i, p_j) \in M$ and $f(s_i, p_j) = 0$ otherwise.
$\forall\ (p_j, l_k) \in E_3$, $f(p_j, l_k) = c'_j$ where $c'_j = |M(p_j)|$
$\forall\ (l_k, v_t) \in E_4$, $f(l_k, v_t) = d'_k$ where $d'_k = |M(l_k)|$

We define a student s_i to be *exposed* if $f(v_s, s_i) = 0$ meaning that there is no flow through s_i. Similarly we define a project p_j to be *exposed* if $f(p_j, l_k) < c_j$ and $f(l_k, v_t) < d_k^+$ where $l_k = l(p_j)$.

Let M be a matching of size k in I. We say that M is a *greedy k-matching* if there is no other matching M' such that $|M'| = k$ and $\rho(M') \succ_L \rho(M)$. If k is the size of a maximum cardinality matching in I, we call M a *greedy maximum matching* in I. Also we say that M is a *generous k-matching* if there is no other

students' preferences:	lecturers' offerings:
$s_1 : p_1 \quad p_2 \quad p_3$	$l_1 : \{p_1, p_2\}$
$s_2 : p_1$	$l_2 : \{p_3\}$
$s_3 : p_2 \quad p_3$	project capacities: $c_1 = 1, c_2 = 1, c_3 = 1$
	lecturer capacities: $d_1 = 2, d_2 = 1$

Fig. 1. A SPA instance I

matching M' such that $|M'| = k$ and $\rho(M') \prec_R \rho(M)$. If k is the size of a maximum cardinality matching in I, we call M a *generous maximum matching* in I.

Figure 1 shows a sample SPA instance with greedy and generous maximum matchings $M_1 = \{(s_1, p_3), (s_2, p_1), (s_3, p_2)\}$ and $M_2 = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$ respectively.

3 Greedy maximum matchings in SPA

In this section we present the algorithm GREEDY-MAX-SPA for finding a greedy maximum matching given a SPA instance. The algorithm is based on the general Ford-Fulkerson algorithm for finding a maximum flow in a network [4]. We obtain maximum profile augmenting paths by adopting techniques used in the bipartite matching approach for finding a greedy maximum matching in HA [9] and CHA [17].

The GREEDY-MAX-SPA algorithm shown in Algorithm 1 takes in a SPA instance I as input and returns a greedy maximum matching M in I. A flow network $N(I) = \langle G, c \rangle$ is constructed as described in Section 2. Given a flow f in $N(I)$ that yields a greedy k-matching $M(f)$ in I, if k is not the size of a maximum flow in $N(I)$, we seek to find a maximum profile augmenting path P with respect to f in $N(I)$ such that the new flow f' obtained by augmenting f along P yields a greedy $(k+1)$-matching $M(f')$ in I. Lemmas 1 and 2 show the correctness of this approach. We firstly show that if k is smaller than the size of a maximum flow in $N(I)$ then such a path is bound to exist.

Lemma 1 *Let I be an instance of* SPA *and let η denote the size of a maximum matching in I. Let k $(1 \leq k < \eta)$ be given and suppose that M_k is a greedy k-matching in I. Let $N = N(I)$ and $f = f(M_k)$. Then there exists an augmenting path P with respect to f in N such that if f' is the result of augmenting f along P then $M_{k+1} = M(f')$ is a greedy $(k+1)$-matching in I.*

Lemma 2 *Let f be a flow in N and let $M_k = M(f)$. Suppose that M_k is a greedy k-matching. Let P be a maximum profile augmenting path with respect to f. Let f' be the flow obtained by augmenting f along P. Now let $M_{k+1} = M(f')$. Then M_{k+1} is a greedy $(k+1)$-matching.*

The GET-MAX-AUG algorithm shown in Algorithm 2 accepts a flow network $N(I)$ and flow f as input and finds an augmenting path of maximum profile

Algorithm 1 GREEDY-MAX-SPA

Require: SPA instance I;
Ensure: return matching M;
1: define flow network $N(I) = \langle G, c \rangle$;
2: define empty flow f;
3: **loop**
4: $P = $ GET-MAX-AUG$(N(I), f)$;
5: **if** $P \neq null$ **then**
6: augment f along P;
7: **else**
8: return $M(f)$;

relative to f or reports that none exists. The latter case implies that $M(f)$ is already a greedy maximum matching. The method consists of three phases: an initialisation phase (lines 1 -11), the main phase which is a loop containing two other loops (lines 12 - 27) and a final phase (lines 28 - 35) where the augmenting path is generated and returned.

For each project p_j the GET-MAX-AUG method maintains a variable $\rho(p_j)$ describing the profile of a partial augmenting path P' from some exposed student to p_j. It also maintains, for every project $p_j \in \mathcal{P}$, a pointer $pred(p_j)$ to the student or lecturer preceding p_j in P'. For every lecturer $l_k \in \mathcal{L}$ a pointer $pred(l_k)$ is also used to refer to any project preceding l_k in P'. Thus the final augmenting path produced will pass through each lecturer or project at most once. The initialisation phase of the method involves setting all *pred* pointers to **null** and ρ profiles to B_R^-. Next, the method seeks to find, for each project p_j, a partial augmenting path $((v_s, s_i), (s_i, p_j))$ from the source, through an exposed student s_i to p_j should one exist. In the presence of multiple paths satisfying this criterion, the path with the best profile (w.r.t. $\succ_L$) is selected. The variables $pred(p_j)$ and $\rho(p_j)$ are updated accordingly. Thus at the end of this phase $\rho(p_j)$ indicates the maximum profile of an augmenting path of length 2 via some exposed student to p_j should one exist. If such a path does not exist then $\rho(p_j)$ and $pred(p_j)$ retain their initial values of B_R^- and **null** respectively.

In the main phase, the algorithm then runs $|f|$ iterations, at each stage attempting to increase the quality (w.r.t. $\succ_L$) of the augmenting paths described by the ρ profiles. Each iteration runs two loops. Each loop identifies cases where the flow through one edge in the network can be reduced in order to allow the flow through another to be increased while improving the profile of the projects involved. In both loops, the decision on whether to switch the flow between candidate edges is made based on an edge relaxation operation similar to that used in the Bellman-Ford algorithm for solving the single source shortest path problem in which edge weights may be negative. In the first loop, we seek to evaluate the gain that may be derived from switching the flow through a student from one project to another. Given an edge (s_i, p_k) with a flow of 1 in f and edge (s_i, p_j) with no flow in f, we define σ to be the resulting profile of p_j if the partial augmenting path ending at p_k is to be extended (via s_i) to p_j.

Thus σ will become the new value of $\rho(p_j)$ should this extension take place. If $\sigma \succ_L \rho(p_j)$ (i.e. if the proposed profile is better than the current one), we extend the augmenting path to p_j and update $\rho(p_j) = \sigma$ and $pred(p_j) = s_i$.

In the second loop, we seek to evaluate the gain that may be derived from switching flow to some lecturer from one project to another. Given a lecturer l_k, let $P'_k \subseteq P_k$ be the set of projects offered by l_k with positive outgoing flow and $P''_k \subseteq P_k$ be the set of projects offered by l_k that are undersubscribed in $M(f)$. Then we seek to determine if an improvement can be obtained by switching a unit of flow from some project $p_j \in P'_k$ to some other project $p_m \in P''_k$. This is achieved by comparing the $\rho(p_j)$ and $\rho(p_m)$ profiles and updating $\rho(p_j) = \rho(p_m)$, $pred(p_j) = l_k$ and $pred(l_k) = p_m$ if $\rho(p_m) \succ_L \rho(p_j)$ where $\rho(p_m)$ represents the profile of a partial augmenting path that does not already pass through l_k (i.e., $pred(p_m) \neq l_k$). This means that the partial augmenting path ending at p_m can be extended further (via l_k) to p_j while improving its profile. The intuition is that, after augmenting along such a path, p_m gains an extra student while p_j loses one.

During the final phase, we iterate through all exposed projects and find the one with the largest profile with respect to $\succ_L$ (say p_q). An augmenting path is then constructed through the network using the *pred* values of the projects and lecturers and the matched edges in $M(f)$ starting from p_q. The generated path is returned to the calling algorithm. If no exposed project exists, the method returns **null**. We next show that GET-MAX-AUG method produces such a maximum profile augmenting path in N with respect to f should one exist.

Lemma 3 *Given a* SPA *instance I, let f be a flow in $N = N(I)$ where $k = |f|$ is not the size of a maximum matching in I and $M(f)$ is a greedy k-matching in I. Algorithm* GET-MAX-AUG *finds a maximum profile augmenting path in N with respect to f.*

From Lemmas 1, 2 and 3, we can conclude that the algorithm GREEDY-MAX-SPA finds a greedy maximum matching given a SPA instance. Concerning the complexity of the algorithm, the main loop calls GET-MAX-AUG η times where η is the size of a maximum cardinality matching in I. The first phase of GET-MAX-AUG performs $O(m_2)$ profile comparison operations and $O(n_3)$ initialisation steps for the lecturer *pred* values where $m_2 = |E_2|$, $n_3 = |\mathcal{L}|$, and each profile comparison step requires $O(R)$ time. The loop in the main phase of GET-MAX-AUG runs k times where k is the value of the flow obtained at that time. The first and second loops perform $O(m_2)$ and $O(n_2)$ relaxation steps respectively where $n_2 = |\mathcal{P}|$ and each relaxation step requires $O(R)$ time to compare profiles. The final phase of the algorithm performs $O(n_2)$ profile comparisons, each also taking $O(R)$ time. Thus the overall time complexity of the GET-MAX-AUG method is $O(m_2R + n_3 + kR(m_2 + n_2) + n_2R) = O(kR(m_2))$. Thus the overall time complexity of the GREEDY-MAX-SPA algorithm is $O(n_1^2 m_2 R)$. A straightforward refinement of the algorithm can be made by observing that if no profile is updated during an iteration of the main loop, then no further profile improvements can be made and we can terminate the main loop at this point. We conclude with the following theorem.

Algorithm 2 GET-MAX-AUG (method for GREEDY-MAX-SPA)

Require: flow network $N(I) = \langle G, c\rangle$ where $G = (V, E)$, flow f where $M(f)$ is a greedy $|f|$-matching;

/* initialisation */
for project $p_j \in \mathcal{P}$ **do**
  $\rho(p_j) = B_R^-$;
  $pred(p_j) =$ **null**;
  for each exposed student $s_i \in S$ such that $p_j \in A_i$ **do**
    $\sigma = O_R + rank(s_i, p_j)$;
    if $\sigma \succ_L \rho(p_j)$ **then**
      $\rho(p_j) = \sigma$;
      $pred(p_j) = s_i$;
for lecturer $l_k \in \mathcal{L}$ **do**
  $pred(l_k) =$ **null**;
/* main phase */
for $1...|f|$ **do**
  /* first loop */
  for each $(s_i, p_j) \in E$ where $f(s_i, p_j) = 0$ and $f(s_i, p_k) = 1$ for some $p_k \in A_i$ **do**
    $\sigma = \rho(p_k) - rank(s_i, p_k) + rank(s_i, p_j)$;
    if $\sigma \succ_L \rho(p_j)$ **then**
      $\rho(p_j) = \sigma$; $pred(p_j) = s_i$;
  /* second loop */
  for each lecturer $l_k \in \mathcal{L}$ **do**
    $\sigma = B_R^-$; $p_z =$ **null**;
    for each project $p_m \in P_k$ such that $l(p_m) = l_k \wedge f(p_m, l_k) < c_m$ **do**
      if $\rho(p_m) \succ_L \sigma$ **then**
        $\sigma = \rho(p_m)$; $p_z = p_m$;
    for each project $p_j \in P_k$ such that $l(p_j) = l_k \wedge f(p_j, l_k) > 0 \wedge p_j \neq p_z$ **do**
      if $\sigma \succ_L \rho(p_j)$ **then**
        $\rho(p_j) = \sigma$; $pred(p_j) = l_k$; $pred(l_k) = p_z$;
/* final phase */
$\rho = \max_{\succ_L}(\{B_R^-\} \cup \{\rho(p_j) : p_j \in P \text{ is exposed}\})$;
if $\rho \succ_L B_R^-$ **then**
  $p_q = \arg\max_{\succ_L}(\{B_R^-\} \cup \{\rho(p_j) : p_j \in P \text{ is exposed}\})$;
  $Q =$ path obtained by following $pred$ values and matched edges in $M(f)$ from p_q to an exposed student;
  return $\langle v_s\rangle$ ++ $reverse(Q)$ ++ $\langle l(p_q), v_t\rangle$; /*++ denotes concatenation*/
else
  return **null**;

Theorem 4 *Given a* SPA *instance I, a greedy maximum matching in I can be obtained in $O(n_1^2 R m_2)$ time.*

4 Generous maximum matchings in SPA

Analogous to the case for greedy maximum matchings, generous maximum matchings can also be found by modelling SPA as a network flow problem. Given a SPA instance I we define the following terms relating to partial augmenting paths

in $N(I)$. For each project $p_j \in \mathcal{P}$, we define the *minimum profile of a partial augmenting path* from v_s through an exposed student to p_j with respect to $\prec_R$, denoted $\Phi'(p_j)$, as follows: $\Phi'(p_j) = \min_{\prec_R}\{\rho(P') : P'$ is a partial augmenting path from v_s to $p_j\}$.

If a partial augmenting path P' ending at project p_j can be extended to an augmenting path P by adding edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ then such an augmenting path is called a *minimum profile augmenting path* if $\rho(P) = \min_{\prec_R}\{\Phi'(p_j) : p_j \in \mathcal{P}\}$. A similar approach to that used to find a greedy maximum matching can be adopted in order to find a generous maximum matching. The main Greedy-max-spa algorithm will remain unchanged (we will call it Generous-max-spa for convenience) as the intuition remains to successively find larger generous k-matchings until a generous maximum matching is obtained. We however make slight changes to the Get-max-aug algorithm in order to find a minimum profile augmenting path in the network should one exist (the resulting algorithm is then known as Get-Min-Aug). The changes are as follows. (i) We replace all occurrences of left domination $\succ_L$ with right domination $\prec_R$. (ii) We also replace all occurrences of negative infinity profile B_R^- with a positive infinity profile B_R^+. (iii) Finally we replace both max functions (in lines 29 and 31) with the min function. Analogous statements and proofs of Lemmas 1, 2 and 3 exist in this context. Thus we may conclude with the following theorem concerning the Generous-max-spa algorithm.

Theorem 5 *Given a* spa *instance I, a generous maximum matching in I can be obtained in $O(n_1^2 R m_2)$ time.*

5 Lecturer lower quotas

We have so far considered a spa model in which each lecturer l_k has an upper quota. In this section we discuss how the algorithm presented above can be modified to allow lecturer lower quotas. We call this extension the *Student/Project problem with Lecturer lower quotas* (spa-l). In an instance I of spa-l, each lecturer l_k now additionally has a lower quota $d_k^-(I)$ (it will be helpful to indicate specific instances to which these lower bounds refer within the notation). We assume that $d_k^-(I) \geq 0$ and $d_k^+(I) \geq \max\{d_k^-(I), 1\}$. In the spa-l context, our definition of a matching as presented in Section 2 needs to be tightened slightly. A *constrained matching* is a matching M in the spa context with the additional property that, for each lecturer l_k, $|M(l_k)| \geq d_k^-(I)$. We seek to find greedy and generous maximum constrained matchings should they exist.

Let I be a spa-l instance. Also let I' be a spa instance constructed from I by setting $d_k^-(I') = 0$ and $d_k^+(I') = d_k^-(I)$ for each lecturer l_k. Firstly we find a greedy maximum matching M' in I' using the Greedy-max-spa algorithm. If $f' = f(M')$ is not a saturating flow (i.e., one in which all edges $(l_k, v_t) \in E_4$ are saturated), then I admits no constrained matching. Otherwise we augment f' in $N(I)$ by calling Greedy-Max-Spa on I, changing line 2 so that flow f is assigned to be f' initially. We continuously augment the flow until no augmenting path exists. The matching $M = M(f)$ obtained from the resulting flow f is a

greedy maximum constrained matching in I. Generous maximum constrained matchings can also be found by using GENEROUS-MAX-SPA and GET-MIN-AUG instead of GREEDY-MAX-SPA and GET-MAX-AUG respectively.

Theorem 6 *Let I be a* SPA-L *instance. Each of the problems of finding a greedy or generous maximum constrained matching, or reporting that no such matching exists, can be solved in $O(n_1^2 R m_2)$ time.*

References

1. Abraham, D. J.: Algorithmics of two-sided matching problems. Master's thesis, University of Glasgow, Department of Computing Science. (2003)
2. Abraham, D. J., Irving, R.W., Manlove, D.F.: Two algorithms for the Student-Project allocation problem. Journal of Discrete Algorithms. 5(1), 79–91 (2007)
3. Abu El-Atta, A.H., Moussa, M.I.: Student project allocation with preference lists over (student,project) pairs. In Proceedings of ICCEE 09: the 2nd International Conference on Computer and Electrical Engineering. 375–379 (2009)
4. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press. (1962)
5. Gusfield, D., Irving, R.W.: The Stable Marriage Problem: Structure and Algorithms. MIT Press. (1989)
6. Huang, C.-C., Kavitha, T., Mehlhorn, K., Michail, D.: Fair matchings and related problems. In Proceedings of FSTTCS 2013: Foundations of Software Technology and Theoretical Computer Science. 24, 339–350 (2013)
7. Hylland, A., Zeckhauser, R.: The efficient allocation of individuals to positions. Journal of Political Economy. 87(2), 293–314 (1979)
8. Irving, R.W.: Greedy matchings. Technical Report TR-2003-136, University of Glasgow, Department of Computing Science. (2003)
9. Irving, R.W.: Greedy and generous matchings via a variant of the Bellman-Ford algorithm. Unpublished manuscript. (2006)
10. Irving, R.W., Kavitha, T., Mehlhorn, K., Michail, D., Paluch, K.: Rank-maximal matchings. ACM Transactions on Algorithms. 2(4), 602–610 (2006)
11. Iwama, K., Miyazaki, S., Yanagisawa, H.: Improved approximation bounds for the student-project allocation problem with preferences over projects. Journal of Discrete Algorithms. 13, 59–66 (2012)
12. Kwanashie, A., Irving, R.W., Manlove, D.F., Sng, C.T.S.: Profile-based optimal matchings in the Student/Project Allocation problem. CoRR Technical Report 1403.0751. Available from `http://arxiv.org/abs/1403.0751`. (2014)
13. Manlove, D.F.: Algorithmics of Matching Under Preferences. World Scientific. (2013)
14. Manlove, D.F., O'Malley, G.: Student project allocation with preferences over projects. Journal of Discrete Algorithms. 6, 553–560 (2008)
15. Mehlhorn, K., Michail, D.: Network problems with non-polynomial weights and applications. Unpublished manuscript. (2006)
16. Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Operations Research. 41(2), 338-350 (1993)
17. Sng, C.T.S.: Efficient Algorithms for Bipartite Matching Problems with Preferences. PhD thesis, University of Glasgow, Department of Computing Science. (2008)
18. Zelvyte, M.: The Student-Project Allocation problem: a network flow model. Honours project dissertation, University of Glasgow, School of Mathematics. (2014)
19. Zhou, L.: On a conjecture by Gale about one-sided matching problems. Journal of Economic Theory. 52(1), 123–135 (1990)