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 \(r\)th component indicates how many students have their \(r\)th-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.

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 \[\text{[1].}\] Moreover the potentially large number of students and projects involved in these schemes motivates the need to discover efficient algorithms for finding optimal matchings.

\(*\)Supported by Engineering and Physical Sciences Research Council grant EP/K010042/1.

\(†\)Work done while at the School of Computing Science, University of Glasgow.
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) [13, 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 [4]. 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 [12].

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 [12]. 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 \( r \)th component indicates the number of students obtaining their \( r \)th-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 \( x_r \) taken over all components \( r \) of the profile, where \( x_r \) is the \( r \)th 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., [12, 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 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 \( R \)th-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...
This is a variant of the well-studied House Allocation problem (HA) \cite{7,17} 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 \cite{6,9,15,14}. 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 \cite{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 finding 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 \cite{1,16} 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 \cite{16} 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 can find rank maximal, minimum cost maximum, generous maximum and greedy maximum matchings in a SPA instance. However this approach involves assigning exponentially large edge weights (see, e.g., \cite[p.405]{12}), which may be computationally infeasible for larger problem instances due to floating point inaccuracies in dealing with such high numbers. For example for larger SPA instances involving say, 1000 students each ranking 20 projects in order of preference, edge weights could potentially be in the order of \(10^6 20\). Since the flow algorithms would require comparing these edge weights, floating point precision errors could easily cause them to fail in practice.

\subsection{Our contribution}

In this paper we present efficient algorithms for finding optimal matchings to SPA problems based 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. Our algorithms run in \(O(n_1^2 R^* (m_2 + n_2^2))\) time where \(n_1\) and \(n_2\) are the numbers of students and projects respectively, \(R^*\) is the maximum rank of any student in the matching and \(m_2\) is the sum of all the students’ preference list lengths. 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. 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.
2 Preliminary definitions

An instance $I$ of the spa problem consists of a set $S$ of students, a set $P$ of projects and a set $L$ of lecturers. Each student $s_i$ ranks a set $A_i \subseteq P$ of projects that she considers acceptable in order of preference. This preference list of projects may contain ties. Each project $p_j \in P$ has an upper quota $c_j$ indicating the maximum number of students that can be assigned to it. Each lecturer $l_k \in L$ offers a set of projects $P_k \subseteq P$ and has an upper quota $d_k^L$ 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 $P$. If project $p_j \in P_k$, then we denote $l_k = l(p_j)$.

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

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

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 $p_j \in P$ and $|M(l_k)| \leq d_k^L$ for all lecturers $l_k \in L$.

Given a student $s_i$ and a project $p_j \in A_i$, we define $\text{rank}(s_i, p_j)$ as $1 + \# \text{projects that } s_i \text{ prefers to } p_j$. 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 $(x_1, x_2, \ldots, x_R)$ 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 $\text{rank}(s_i, p_j) = r$. Let $\alpha = (x_1, x_2, \ldots, x_R)$ and $\sigma = (y_1, y_2, \ldots, y_R)$ be any two profiles. We define the empty profile $O_R = (0, 0, \ldots, 0)$ where $0_r = 0$ for all $r \ (1 \leq r \leq R)$. We also define the negative infinity profile $B^-_R = (b_1, b_2, \ldots, b_R)$ where $b_r = -\infty$ for all $r \ (1 \leq r \leq R)$ and the positive infinity profile $B^+_R = (b_1, b_2, \ldots, b_R)$ where $b_r = +\infty$ for all $r \ (1 \leq r \leq R)$. We define the sum of two profiles $\alpha$ and $\sigma$ as $\alpha + \sigma = (x_1 + y_1, x_2 + y_2, \ldots, x_R + y_R)$. Given any $q \ (1 \leq q \leq R)$, we define $\alpha + q = (x_1, \ldots, x_{q-1}, x_q + 1, x_{q+1}, \ldots, x_R)$. We define $\alpha - q$ in a similar way.

We define the total order $\succ_L$ on profiles as follows. We say $\alpha \succ_L \sigma$ 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 $\alpha \prec_R \sigma \ (\alpha \prec_R \sigma)$ 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 $\alpha \preceq_R \sigma \ (\alpha \preceq_R \sigma)$ 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 $N(I) = (G, c)$ where $G = (V, E)$ is a directed graph and $c$ is a non-negative capacity function $c : E \rightarrow \mathbb{R}^+$ 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 $V = \{v_s, v_t\} \cup S \cup P \cup L$ and $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where $E_1 = \{(v_s, s_i) : s_i \in S\}$, $E_2 = \{(s_i, p_j) : s_i \in S, p_j \in A_i\}$, $E_3 = \{(p_j, l_k) : p_j \in P, l_k = l(p_j)\}$ and $E_4 = \{(l_k, v_t) : l_k \in L\}$. 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 $c(l_k, v_t) = d_k^L$ 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 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 $\rho(P')$,
students’ preferences:  
$s_1 : p_1 \ p_2 \ p_3$  
$s_2 : p_1$  
$s_3 : p_2 \ p_3$

lecturers’ offerings:  
$l_1 : \{p_1, p_2\}$  
$l_2 : \{p_3\}$  
project capacities: $c_1 = 1, c_2 = 1, c_3 = 1$  
lecturer capacities: $d_1 = 2, d_2 = 1$

Figure 1: A SPA instance $I$

as follows:

$$
\rho(P') = O_R + \sum\{\text{rank}(s_i, p_j) : (s_i, p_j) \in P' \land f(s_i, p_j) = 0\} \\
- \sum\{\text{rank}(s_i, p_j) : (p_j, s_i) \in P' \land f(s_i, p_j) = 1\}
$$

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 $(l(p_j), v_l)$ where $v_l$ 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 $\rho(P) = \max_{\succ_L} \{\Phi(p_j) : p_j \in P\}$.

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 $c'_j = |M(p_j)|$

$\forall (l_k, v_l) \in E_4, f(l_k, v_l) = d'_k$ 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 $f(l_k, v_l) < d'_k$ 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 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 [15].

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 $N(I) = (G, c)$ 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 $\eta$ 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 $I$.

**Proof** Let $I' = C(I)$ be a new instance of spa obtained from $I$ as follows. Firstly we add all students in $I$ to $I'$. Next, for every project $p_j \in P$, we add $c_j$ clones $p_j^1, p_j^2, \ldots, p_j^c_j$ to $I'$. We then add all lecturers in $I$ to $I'$. If the student-project pair $(s_i, p_j)$ is in $I$, we add $(s_i, p_j^r)$ to $I'$ for all $1 \leq r \leq c_j$. If the project-lecturer pair $(p_j, l_k)$ is in $I$, we add $(p_j^r, l_k)$ to $I'$ for all $1 \leq r \leq c_j$. Also if $\text{rank}(s_i, p_j) = t$, we set $\text{rank}(s_i, p_j^r) = t$ for all $r$ ($1 \leq r \leq c_j$). Let $G'$ be the underlying graph in $I'$ involving only the student and project clones. With respect to the matching $M_k = M(f)$, we construct a cloned matching $C(M_k)$ in $I'$ as follows. If project $p_j$ is assigned $x_j$ students $s_{q_1}, s_{q_2}, \ldots, s_{q_{x_j}}$ in $M_k$ we add $(s_{q_r}, p_j^r)$ to $C(M_k)$ for all $1 \leq r \leq x_j$. Hence $C(M_k)$ is a greedy $k$-matching in $I'$.

Let $M_{k+1}'$ be a greedy $(k + 1)$-matching in $I$ (this exists because $k < \eta$). Then $C(M_{k+1}')$ is a greedy $(k + 1)$-matching in $I'$. Let $X = C(M_k) \oplus C(M_{k+1}')$. Then each connected component of $X$ is either (i) an alternating cycle, (ii) an even-length alternating path or (iii) an odd-length alternating path in $G'$ (with no restrictions on which matching the end edges belong to). The aim is to show that, by eliminating a subset of $X$, we are left with a set of connected components which can be transformed into a single augmenting path with respect to $f(C(M_k))$ in $N(I')$ and subsequently a single augmenting path with respect to $f(M_k)$ in $N(I)$.

**Eliminating connected components of $X$:** Suppose $D \subseteq X$ is a type (i) connected component of $X$ or a type (ii) connected component of $X$ whose end vertices are students (we may call this a type (ii)(a) component). Suppose also that $\rho(D \cap C(M_k')) \succ_L \rho(D \cap C(M_k))$. A new matching $C(M_k'$) in $G'$ of cardinality $k$ can be created from $C(M_k)$ by replacing all the $C(M_k)$-edges in $D$ with the $C(M_{k+1}')$-edges in $D$ (i.e., by augmenting $C(M_k)$ along $D$). Since the upper quota constraints of the lecturers involved are not violated after creating $C(M_k')$ from $C(M_k)$, it follows that $C(M_k')$ is also a valid spa matching in $I'$. Moreover $\rho(C(M_k')) \succ_L \rho(C(M_k))$ which is a contradiction to the fact that $C(M_k)$ is a greedy $k$-matching in $I'$. A similar contradiction (to the fact that $C(M_k')$ is a greedy $(k + 1)$-matching in $I'$) exists if we assume $\rho(D \cap C(M_k)) \succ_L \rho(D \cap C(M_{k+1}'))$. Thus $\rho(D \cap C(M_k')) = \rho(D \cap C(M_k))$.

Form the argument above all type (i) and type (ii)(a) connected components of $X$ do not contribute to a change in the size or profile as we augment from $C(M_k)$ to $C(M_k')$ or vice versa. In fact, this is true for any even-length connected component of $X$ which does not cause lecturer upper quota constraints to be violated as we augment from $C(M_k)$ to $C(M_{k+1}')$ or vice versa. The claim can further be extended to certain groups of connected components which, when considered together, (i) have equal numbers of $C(M_k)$ and $C(M_{k+1}')$ edges and (ii) do not cause lecturer upper quota constraints to be violated as we augment from $C(M_k)$ to $C(M_{k+1}')$ or vice versa. In all these cases, it is possible to eliminate such components (or groups of
components) from consideration. Using the above reasoning, we begin by eliminating all type (i) and type (ii)(a) connected components of $X$.

Let $D$ be the union of all the edges in type (i) and type (ii)(a) connected components of $X$. Let $X' = X \setminus D$. Then it follows that $X' = C(M_k) \oplus C(M_k')$ for some greedy $(1)$-matching $C(M_{k+1})$ in $I'$ which can be constructed by augmenting $C(M_{k+1})$ along all type (i) and type (ii)(a) components of $X$. Thus $X'$ contains (1) even-length alternating paths whose end vertices are project clones (we call these type (ii)(b) paths), (2) odd-length alternating paths whose end edges are in $C(M_k)$ (we call these type (iii)(a) paths) and (3) odd-length alternating paths whose end edges are in $C(M_{k+1})$ (we may call these type (iii)(b) paths). Although these alternating paths are disjoint, there are special cases where two alternating paths in $X'$ may be joined together by pairing their end project clone vertices.

**Joining alternating paths:** Consider some lecturer $l_q$ with upper quota $d_q$. Let $X_q = \{(s_i, p_j^e) \in C(M_k) : l_q = l(p_j^e) \land p_j^e \text{ is unmatched in } C(M_{k+1})\}$ and $x_q = |X_q|$. Thus $X_q$ is the set of end edges incident to project clones belonging to a subset of the type (ii)(b) and type (iii)(a) paths in $X'$. Let $Y_q = \{(s_i, p_j^e) \in C(M_{k+1}) : l_q = l(p_j^e) \land p_j^e \text{ is unmatched in } C(M_k)\}$ and $y_q = |Y_q|$. Thus $Y_q$ is the set of end edges incident to project clones belonging to a subset of the type (ii)(b) and type (iii)(a) paths in $X'$. Let $Z_q = \{p_j^p : l_q = l(p_j^e) \land p_j^e \text{ is matched in } C(M_k) \text{ and } z_q = |Z_q|\}$. Thus $x_q + z_q \leq d_q$ and $y_q + z_q \leq d_q$. If $l_q$ is full in $C(M_k)$ then $x_q \geq y_q$. Also if $l_q$ is full in $C(M_{k+1})$ then $y_q \geq x_q$. Finally if $l_q$ is full in both $C(M_k)$ and $C(M_{k+1})$ then $x_q = y_q$. Thus considering lecturer $l_q$, for each alternating path in $X'$ with end edge $(s_i, p_j^e) \in Y_q$, if $l_q$ is full in $C(M_k)$ there exists a unique complementary alternating path in $X'$ with end edge $(s_i', p_{j'}^e) \in X_q$. Such complementary alternating paths can be joined together by pairing both end projects clones thus forming a project pair $(p_j^p, p_{j'}^p)$ at $l_q$. Thus we may form a set of unique project pairs involving project clones from all $(y_q)$ edges in $Y_q$ and $y_q$ edges in $X_q$. Similarly for each alternating path in $X'$ with end edge $(s_i, p_j^e) \in X_q$, if $l_q$ is full in $C(M_{k+1})$ there exists a unique complementary alternating path in $X'$ with end edge $(s_i', p_{j'}^e) \in Y_q$. Again such complementary alternating paths can be joined together by pairing both end project clones thus forming a project pair $(p_j^p, p_{j'}^p)$ at $l_q$. Thus we may form a set of unique project pairs involving project clones from all $(x_q)$ edges in $X_q$ and $x_q$ edges in $Y_q$. Finally if $l_q$ is full in both $C(M_k)$ and $C(M_{k+1})$ then every alternating path in $X'$ with end edge in $X_q$ has a unique complementary alternating path in $X'$ with end edge in $Y_q$ thus forming a $1 \rightarrow 1$ pairing of project clones from edges in $X_q$ and project clones from edges in $Y_q$. The resulting path (which we call a compound path) may be regarded as a single path along which $C(M_k)$ or $C(M_{k+1})$ may be augmented. We begin such pairings by considering type (iii)(b) paths in $X'$.

Consider some type (iii)(b) path $D'$ in $X'$. Let $(s_i, p_j^e)$ be the end edge of $D'$ incident to a project clone that is unmatched in $C(M_k)$. Then $(s_i, p_j^e) \in Y_q$ where $l_q = l(p_j^e)$. The lecturer $l_q$ can either be undersubscribed or full in $C(M_k)$. If $l_q$ is undersubscribed in $C(M_k)$ we call $D'$ an augmenting type (iii)(b) path. On the other hand if $l_q$ is full in $C(M_k)$ then $x_q \geq y_q$ (from the argument above) and so there exists a unique edge $(s_i, p_{j'}^e) \in X_q$ which is the end edge of some path $D'_1$ in $X'$. We can view $D' \cup D'_1$ as a single compound path joined together by pairing $p_j^p$ with $p_{j'}^p$.

If $D'_1$ is a type (iii)(a) path then $D' \cup D'_1$ has an even number of edges with both end vertices being students. Also the the number of students assigned to each lecturer remains the same if we augment $C(M_k)$ or $C(M_{k+1})$ along $D' \cup D'_1$ (i.e., the lecturer upper quota constraints are not violated). Thus the argument presented above for type (i) and type (ii)(a) components holds and $D' \cup D'_1$ can be eliminated from $X'$. On the other hand if $D'_1$ is a type (ii)(b) path where the other end vertex is project clone $p_{j''}^p$ then $D' \cup D'_1$ has an odd number of edges. It may be that $l(p_{j''}^e)$ is undersubscribed in $C(M_k)$ in which case we denote $D' \cup D'_1$ as a compound
type (iii)(b) path (in general, we refer to a compound type (iii)(b) path as one comprising an augmenting type (iii)(b) path joined together with zero or more type (ii)(b) paths). However if \( l(p'_{jr}) \) is full in \( C(M_k) \) then \( D' \cup D'_1 \) must again be joined together with either a type (iii)(a) or a type (ii)(b) path (not previously considered due to the presence of unique project clone pairings). Since there are a finite number of such paths in \( X' \) we are bound to end up with a chain of joined together paths that have either (i) an odd number of edges with the end project clone vertex offered by an undersubscribed lecturer in \( C(M_k) \) (a compound type (iii)(b) path) or (ii) an even number of edges with both end vertices as students (and so can be eliminated). We will deal with compound type (iii)(b) paths later in the proof.

Next consider each remaining type (iii)(a) path \( D'' \) in \( X' \). Let \( p'_{jr} \) be the end project vertex. The lecturer \( l_q = l(p'_{jr}) \) can either be undersubscribed or full in \( C(M'_{k+1}) \). As in the case for type (iii)(b) paths, if \( l_q \) is undersubscribed in \( C(M'_{k+1}) \) we call \( D'' \) an augmenting type (iii)(a) path. On the other hand if \( l(p'_{jr}) \) is full in \( C(M'_{k+1}) \) then \( y_q \geq x_q \) and so there exists a unique edge \((s_i, p'_{jr}) \in Y_q \) which is the end of some path \( D''_1 \) in \( X' \). Using the same argument presented for the type (iii)(b) paths above, \( D' \cup D'_1 \) may either yield a compound type (iii)(a) path or may be extended further (in general, we refer to a compound type (iii)(a) path as one comprising an augmenting type (iii)(a) path joined together with zero or more type (ii)(b) paths). As we extend the compound path by pairing end edges in \( X_q \) and \( Y_q \) for a lecturer \( l_q \) who is full in \( C(M'_{k+1}) \), if an even-length path is obtained (with both end vertices as students), then the arguments made above hold and it can be eliminated from consideration. At the end of this phase we are left with compound type (iii)(a) paths in \( X' \). We will again deal with these paths later on in the proof.

We’ve so far seen how various types of even-length paths (or combinations of joined together paths) can be eliminated from consideration as they do not affect the size or profile as we augment from a greedy \( k \)-matching to a greedy \((k+1)\)-matching or vice versa. We now consider any remaining type (ii)(b) paths \( D''' \) in \( X' \). Consider both end edges \((s_i, p''_{jr}) \in C(M_k) \) and \((s_i, p''_{jr}) \in C(M'_{k+1}) \). If \( l(p''_{jr}) \) is undersubscribed in \( C(M'_{k+1}) \) and \( l(p''_{jr}) \) is undersubscribed in \( C(M_k) \) then lecturer upper quota constraints are not violated if we augment \( C(M_k) \) or \( C(M'_{k+1}) \) along \( D''' \) and so \( D''' \) can be eliminated from consideration. On the other hand, suppose \( l_q = l(p''_{jr}) \) is full in \( C(M'_{k+1}) \). Then \( y_q \geq x_q \) and so there exists a unique edge \((s_i, p''_{jr}) \in Y_q \) which is the end of some type (ii)(b) path \( D''''_1 \) in \( X' \). Thus \( D''' \cup D''''_1 \) can be considered as a single alternating path joined together by pairing \( p''_{jr} \) with \( p''''_{jr} \). The two end edges of \( D''' \cup D''''_1 \) can then be evaluated and if lecturer upper quota constraints are not violated when augmenting \( D''' \cup D''''_1 \) along \( C(M_k) \) or \( C(M'_{k+1}) \), then \( D''' \cup D''''_1 \) can be eliminated from consideration. Otherwise we can extend \( D''' \cup D''''_1 \) using the same technique presented above. Since there are a finite number of possible extensions, we are bound to end up with an even-length path that does not violate lecturer capacities when we augment along the path in either way and so can be eliminated from consideration. The same argument can be made if \( l_m = l(p''_{jr}) \) is full in \( C(M_k) \).

Eliminating compound paths: At this stage we are left with only compound type (iii)(a) and compound type (iii)(b) paths in \( X' \). These paths, if considered independently decrease and increase the size of \( C(M_k) \) by 1 respectively. Since \(|C(M'_{k+1})| = |C(M_k)| + 1 \) then there are \( q \) type (iii)(a) paths and \((q+1) \) type (iii)(b) paths. Consider some compound type (iii)(b) path \( D' \) and some compound type (iii)(a) path \( D'' \). Then we can consider the combined effect of augmenting \( C(M_k) \) or \( C(M'_{k+1}) \) along \( D' \cup D'' \). Suppose that \( \rho((D' \cup D'') \cap C(M'_{k+1})) \geq L \rho((D' \cup D'') \cap C(M_k)) \). A new matching \( C(M''_k) \) in \( G' \) of cardinality \( k \) can be created by augmenting \( C(M_k) \) along \( D' \cup D'' \). Since the upper quota constraints on the lecturers involved are not violated after creating \( C(M''_k) \) from \( C(M_k) \), then \( C(M''_k) \) is also a valid \( k \)-matching in \( G' \). Thus \( \rho(C(M''_k)) \geq L \rho(C(M_k)) \) which is a contradiction to the fact that \( C(M_k) \) is a greedy \( k \)-matching in \( G' \). A similar contradiction (to the fact that \( C(M'_{k+1}) \) is a greedy \((k+1)\)-
matching in $I'$ exists if we assume $\rho((D' \cup D'') \cap C(M_k)) \succ_L \rho((D' \cup D'') \cap C(M'\!_{k+1}))$. Thus $\rho((D' \cup D'') \cap C(M'\!_{k+1})) = \rho((D' \cup D'') \cap C(M_k))$. It follows that, considering $D'$ and $D''$ together, the size and profile of the matching is unaffected as augment from $C(M_k)$ to $C(M'\!_{k+1})$ or vice versa and so both $D'$ and $D''$ can be eliminated from consideration.

**Generating an augmenting path in $N(I)$:** Once all these eliminations have been done, since $|C(M'\!_{k+1})| = |C(M_k)| + 1$ it is easy to see that there remains only one path $P'$ left in $X'$ which is a compound type (iii)(b) path. The path $P'$ can then be transformed to a component $D$ in $G(I)$ (where $G(I)$ is the underlying graph in $I$) by replacing all the project clones $p_j^r$ ($1 \leq r \leq c_j$) in $P'$ with the original project $p_j$. Thus a project may now appear more than once in $D$. A lecturer may also appear more than once in $D$.

Consider some project $p_j \in D$ that appears more than once. Then let $P'' \subset P'$ be the path consisting of edges between the first and last occurrence of the $p_j$ clones in $P'$ ($P''$ corresponds to a collection of cycles belonging to $D$ in $N(I)$ involving $p_j$). Thus $P''$ is of even length and both end projects of $P''$ are clones of the same project. Augmenting $C(M_k)$ or $C(M'\!_{k+1})$ along $P''$ will not violate the lecturer upper quota constraints or affect the size or profile of the matching obtained (again using the same arguments presented above). Thus $P''$ can be eliminated from consideration. Although this potentially breaks $P'$ into two separate paths in $G(I')$ it still remains connected in $G(I)$. Similarly consider some lecturer $l_k \in D$ that appears more than once. Then let $P''' \subset P'$ be the path consisting of edges between the first and last occurrence of the $l_k$ clones in $P'$ ($P'''$ corresponds to a collection type (ii)(b) paths with project clones offered by $l_k$). Thus augmenting $C(M_k)$ or $C(M'\!_{k+1})$ along $P'''$ will not violate the lecturer upper quota constraints or affect the size or profile of the matching obtained (again using the same arguments presented above). Thus $P'''$ can be eliminated from consideration. Doing the above steps continually for all projects and lecturers that occur more than once in $D$ eventually yields a valid path in $G(I)$ in which all nodes are visited only once.

Finally we describe how the path $D$ in $G(I)$, obtained after removing duplicate projects and lecturers, can be transformed to an augmenting path $P$ in $N(I)$ (i.e. we establish the direction of flow from $v_s$ to $v_t$ through $P$ in $N(I)$). Firstly we add the edge $(v_s, s_i)$ to $P$ where $s_i$ is the exposed student in $D$. Next for every edge $(s_{i'}, p_{j'}) \in M_k \cap D$ we add a forward edge $(s_{i'}, p_{j'})$ to $P$. Also for every edge $(s_{i''}, p_{j''}) \in M_k \cap D$ we add a backward edge $(p_{j''}, s_{i''})$ to $P$. Finally we add the edges $(p_j, l(p_j))$ and $(l(p_j), v_t)$ to $P$ where $p_j^r$ is the end project vertex in $D$. Thus $P$ is an augmenting path with respect to $f = f(M_k)$ in $N(I)$ such that if $f'$ is the flow obtained when $f$ is augmented along $P$ then $M(f')$ is a greedy $(k+1)$-matching in $N(I)$. \( \square \)

**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.

**Proof** Suppose for a contradiction that $M_{k+1}$ is not a greedy $(k+1)$-matching. By Lemma 1 there exists an augmenting path $P'$ with respect to $f$ such that if $f'$ is the result of augmenting $f$ along $P'$ then $M_{k+1} = M(f')$ is a greedy $(k+1)$-matching. If $\rho(M'_{k+1}) = \rho(M_{k+1})$ then $M_{k+1}$ is a greedy $(k+1)$-matching, a contradiction to our initial assumption. Now suppose $\rho(M'_{k+1}) \succ_L \rho(M_{k+1})$. Then $M'_{k+1}$ is not a greedy $(k+1)$-matching, a contradiction to Lemma 1. Hence $\rho(M'_{k+1}) \succ_L \rho(M_{k+1})$. Since $\rho(M'_{k+1}) = \rho(M) + \rho(P')$ and $\rho(M_{k+1}) = \rho(M) + \rho(P)$, it follows that $\rho(P') \succ_L \rho(P)$, a contradiction to the assumption that $P$ is a maximum profile augmenting path. \( \square \)

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 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 1-14), the main phase which is a loop
containing two other loops (lines 15 - 34) and a final phase (lines 35 - 43) where the augmenting path is generated and returned.

For each project \( p_j \) the \textsc{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 \( p_j \in P \), a pointer \( \text{pred}(p_j) \) to the student or lecturer preceding \( p_j \) in \( P' \). For every lecturer \( l_k \in L \) a pointer \( \text{pred}(l_k) \) 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 \( \text{pred} \) pointers to \text{null} and \( \rho \) 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. \( >_L \)) is selected. The variables \( \text{pred}(p_j) \) 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 \( \text{pred}(p_j) \) remain \( B^-_R \) and \text{null} respectively.

In the main phase, the algorithm then runs \(|f|\) iterations, at each stage attempting to increase the quality (w.r.t. \( >_L \)) of the augmenting paths described by the \( \rho \) 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 and edge \((s_i, p_j)\) with no flow in \( f \), we define \( \sigma \) 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 \( \sigma \) will become the new value of \( \rho(p_j) \) should this extension take place. If \( \sigma >_L \rho(p_j) \) (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 \( \text{pred}(p_j) = s_i \).

In the second loop, we seek to evaluate the gain that may be derived from switching flow from some lecturer to one project to another. Given a lecturer \( l_k \) who is full in \( M(f) \), 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 \( \rho(p_j) = \rho(p_m) \), \( \text{pred}(p_j) = l_k \) and \( \text{pred}(l_k) = p_m \) if \( \rho(p_m) >_L \rho(p_j) \) where \( \rho(p_m) \) represents the profile of a partial augmenting path that does not already pass through \( l_k \) (i.e., \( \text{pred}(p_m) \neq l_k \)). 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 $\text{pred}$ 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.

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

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

1: /* initialisation */
2: for project $p_j \in P$ do
3: $\rho(p_j) = B_R$;
4: $\text{pred}(p_j) = \text{null}$;
5: for each exposed student $s_i \in S$ such that $p_j \in A_i$ do
6: $\sigma = O_R + \text{rank}(s_i, p_j)$;
7: if $\sigma \succ_L \rho(p_j)$ then
8: $\rho(p_j) = \sigma$; $\text{pred}(p_j) = s_i$;
9: end if
10: end for
11: end for
12: for lecturer $l_k \in L$ do
13: $\text{pred}(l_k) = \text{null}$;
14: end for
15: /* main phase */
16: for $1 \ldots |f|$ do
17: /* first loop */
18: 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
19: $\sigma = \rho(p_k) - \text{rank}(s_i, p_k) + \text{rank}(s_i, p_j)$;
20: if $\sigma \succ_L \rho(p_j)$ then
21: $\rho(p_j) = \sigma$; $\text{pred}(p_j) = s_i$;
22: end if
23: end for
24: /* second loop */
25: for each project $p_j$ where $f(p_j, l_k) > 0$ and $f(l_k, v_t) = d_k^v$ where $l_k = \ell(p_j)$ do
26: for each project $p_m \in P_k$ such that $f(p_m, l_k) < c_m$ and $\text{pred}(p_m) \neq l_k$ do
27: if $\rho(p_m) \succ_L \rho(p_j)$ then
28: $\rho(p_j) = \rho(p_m)$;
29: $\text{pred}(p_j) = l_k$;
30: $\text{pred}(l_k) = p_m$;
31: end if
32: end for
33: end for
34: end for
35: /* final phase */
36: $\rho = \max_{\succ_L} (\{O_r\} \cup \{\rho(p_j) : p_j \in P \text{ is exposed}\})$;
37: if $\rho \succ_L O_r$ then
38: $p_q = \arg \max_{\succ_L} (\{O_r\} \cup \{\rho(p_j) : p_j \in P \text{ is exposed}\})$;
39: $Q = \text{path obtained by following pred values and matched edges in } M(f) \text{ from } p_q \text{ to an exposed student}$;
40: return $\langle v_s \rangle \leftrightarrow \text{reverse}(Q) \leftrightarrow \langle \ell(p_q), v_t \rangle$; /*++ denotes concatenation*/
41: else
42: return null;
43: end if
Lemma 3 Let f be a flow in N where k = |f| is not maximum. Algorithm Get-max-aug finds a maximum profile augmenting path in N with respect to f.

Proof Consider some project p_j in N = (G, c) where G = (V, E). Let Φ_{2k+1}(p_j) be the maximum profile of any partial augmenting path of length ≤ 2k+1 (excluding the first edge from v_s) from an exposed student to p_j. If such a path does not exist then Φ_{2k+1}(p_j) = B_R. Firstly we seek to show that after q iterations of the main loop of Get-max-aug where 0 ≤ q ≤ k, ρ(p_j) ≥ L Φ_{2k+1}(p_j) for every project p_j ∈ P.

We prove this inductively. For the base case, let q = 0. Then Φ_1(p_j) is the maximum profile of any partial augmenting path of length 1 from an exposed student to p_j. Hence, from the initialisation phase of Get-max-aug, ρ(p_j) = Φ_1(p_j).

For the inductive step, assume 1 ≤ q ≤ k and that if the claim is true after the (q - 1)th iteration i.e. \( \rho_{q-1}(p_m) \geq L \Phi_{2q-1}(p_m) \) for any \( p_m \in P \) where \( \rho_{q-1}(p_m) \) denotes the value of \( \rho(p_m) \) after the (q - 1)th iteration, we will show that the claim is true for the qth iteration i.e. \( \rho_q(p_m) \geq L \Phi_{2q+1}(p_m) \). For each project \( p_m \in P \) let \( S'_m = \{ s_i \in S : (s_i, p_j) \in E, f(s_i, p_j) = 0 \} \) and for each lecturer \( l_k \in L \) let \( P'_k = \{ p_m \in P : l_k = l(p_m), f(p_m, l_k) < c_j \} \). For each iteration of the main loop, we perform a relaxation step involving some pair \((s_i, p_j)\) where \( s_i \in S'_j \) and a relaxation step involving some pair \((p_m, l_k)\) where \( p_m \in P'_k \). Consider some project \( p_m \): if there does not exist a partial augmenting path from an exposed student to \( p_m \), of length ≤ 2q + 1 and with a better profile than \( \Phi_{2q-1}(p_m) \), then \( \Phi_{2q+1}(p_m) = \Phi_{2q-1}(p_m) \). Otherwise there exists a partial augmenting path of length 2q + 1 from an exposed student to \( p_m \) with profile \( \Phi_{2q+1}(p_m) \). Such a path must contain a partial augmenting path from an exposed student to some project \( p'_m \) such that:

\[
\Phi_{2q+1}(p_m) = \max_{s_l} \{ \Phi_{2q-1}(p'_m) + \text{rank}(s_i, p_j) - \text{rank}(s_i, p'_{m}) : s_i \in S'_m \wedge f(s_i, p_m) = 1 \}, \{ \Phi_{2q-1}(p'_m) : l_k = l(p_m) \wedge p'_m \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_i) = d_k^+ \} \}
\]

Thus we may also note the following identity involving \( \Phi_{2q+1}(p_m) \):

\[
\Phi_{2q+1}(p_m) = \max_{s_l} \{ \Phi_{2q-1}(p'_m), \{ \Phi_{2q-1}(p'_m) + \text{rank}(s_i, p_j) - \text{rank}(s_i, p'_{m}) : s_i \in S'_m \wedge f(s_i, p_m) = 1 \}, \{ \Phi_{2q-1}(p'_m) : l_k = l(p_m) \wedge p'_m \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_i) = d_k^+ \} \}
\]

From the Get-max-aug algorithm after q iterations of the main loop:

\[
\rho_q(p_m) = \max_{s_l} \{ \rho_{q-1}(p_m), \{ \rho_{q-1}(p'_m) + \text{rank}(s_i, p_j) - \text{rank}(s_i, p'_{m}) : s_i \in S'_m \wedge f(s_i, p_m) = 1 \}, \{ \rho_{q-1}(p'_m) : l_k = l(p_m) \wedge p'_m \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_i) = d_k^+ \} \}
\]

By the induction hypothesis, \( \rho_{q-1}(p_m) \geq L \Phi_{2q-1}(p_m) \). Thus:

\[
\Phi_{2q+1}(p_m) \geq L \max_{s_l} \{ \rho_{q-1}(p_m), \{ \Phi_{2q-1}(p'_m) + \text{rank}(s_i, p_j) - \text{rank}(s_i, p'_{m}) : s_i \in S'_m \wedge f(s_i, p_m) = 1 \}, \{ \Phi_{2q-1}(p'_m) : l_k = l(p_m) \wedge p'_m \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_i) = d_k^+ \} \}
\]

Again by the induction hypothesis, \( \rho_{q-1}(p'_m) \geq L \Phi_{2q-1}(p'_m) \). Thus:

\[
\Phi_{2q+1}(p_m) \geq L \max_{s_l} \{ \rho_{q-1}(p_m), \{ \Phi_{2q-1}(p'_m) + \text{rank}(s_i, p_j) - \text{rank}(s_i, p'_{m}) : s_i \in S'_m \wedge f(s_i, p_m) = 1 \}, \{ \rho_{q-1}(p'_m) : l_k = l(p_m) \wedge p'_m \in P'_k \wedge f(p_m, l_k) > 0 \wedge f(l_k, v_i) = d_k^+ \} \}
\]

Thus after k iterations \( \rho(p_j) > L \Phi_{2k+1}(p_j) \). But any partial augmenting path from an exposed student to \( p_j \) with respect to flow \( f \) can have length at most \( 2k + 1 \) (excluding the edge from \( v_s \)). Thus \( \rho(p_j) = \Phi_{2k+1}(p_j) \) after k iterations.
Finally we show that a partial augmenting path $P'$ (and subsequently a full augmenting path) can be constructed by following the $\text{pred}$ values of projects and lecturers and the matched edges in $M(f)$ starting from some exposed project $p_j$ with the maximum $\rho(p_j)$ profile going through some exposed student and ending at the source $v_s$ (i.e. we show that such a path is continuous and contains no cycle).

Suppose for a contradiction that such a path $P'$ contained a cycle $C$. Then at some step $X$ during the execution of the algorithm, $C$ would have been formed when, for some project $p_j$, either (i) $\text{pred}(p_j)$ was set to some student $s_i$ or (ii) $\text{pred}(p_j)$ was set to some lecturer $l_k$. Let $P''$ be any path in $N(I)$. We may extend our definitions for the profile of a matching and a partial augmenting path to cover the profile of any path in $N(I)$ as follows:

$$\rho(P'') = O_R + \sum\{\text{rank}(s_i, p_j) : (s_i, p_j) \in P'' \cap E_2 \land f(s_i, p_j) = 0\} - \sum\{\text{rank}(s_i, p_j) : (p_j, s_i) \in P'' \cap E_2 \land f(s_i, p_j) = 1\}.$$ 

Considering case (i) let $p_m = M(s_i)$. Also let $\rho'(p_j)$ and $\rho(p_j)$ be the profiles of partial augmenting paths from $v_s$ through some exposed student to $p_j$ before and after step $X$ respectively. Then $\rho(p_j) \succ_L \rho'(p_j)$. Also $\rho(p_j) = \rho(p_m) + \text{rank}(s_i, p_j) - \text{rank}(s_i, p_m)$, i.e., $\rho(p_j) = \rho(p_m) + \rho(P'')$ where $P'' = \{(s_i, p_j), (s_i, p_m)\}$. Since we can also trace a path through all the other projects in $C$ (using $\text{pred}$ values and matched edges) from $p_m$ to $p_j$, it follows that $\rho(p_m) = \rho'(p_j) + \rho(C \setminus \{(s_i, p_j), (s_i, p_m)\})$. Thus $\rho(p_j) = \rho'(p_j) + \rho(C)$. Note that $\rho(C) = \rho(C' \setminus M) - \rho(C' \setminus M)$ and $C' = C \cap E_2$ is the set of edges in $C$ involving only students and projects. As $\rho(p_j) \succ_L \rho'(p_j)$, it follows that $\rho(C' \setminus M) \succ_L \rho(C' \setminus M)$. But since $|C' \setminus M| = |C' \setminus M|$ and lecturer capacities are clearly not violated by the algorithm, a new matching $M' = M \oplus C'$ can be generated such that $\rho(M') \succ_L \rho(M)$ and $|M'| = |M| = |f|$, a contradiction to the fact that $M$ is a greedy $|f|$-matching in $I$.

Considering case (ii) let $p_m = \text{pred}(l_k)$. As before let $\rho'(p_j)$ and $\rho(p_j)$ be the profiles of partial augmenting paths from $v_s$ through some exposed student to $p_j$ before and after step $X$ respectively. Then $\rho(p_j) \succ_L \rho'(p_j)$. Also $\rho(p_j) = \rho(p_m)$. Since we can also trace a path through all the other projects in $C$ (using $\text{pred}$ values and matched edges) from $p_m$ to $p_j$, it follows that $\rho(p_m) = \rho'(p_j) + \rho(C \setminus \{(p_j, l_k), (p_m, l_k)\}) = \rho'(p_j) + \rho(C)$. Thus $\rho(p_j) = \rho'(p_j) + \rho(C)$. Note that $\rho(C) = \rho(C' \setminus M) - \rho(C' \setminus M)$ and $C' = C \cap E_2$ is the set of edges in $C$ involving only students and projects. As $\rho(p_j) \succ_L \rho'(p_j)$, it follows that $\rho(C' \setminus M) \succ_L \rho(C' \setminus M)$. A similar argument to the one presented above shows a contradiction to the fact that $M$ is a greedy $|f|$-matching in $I$. \hfill $\square$

From Lemmas 11 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 $\eta$ times where $\eta$ 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 $\text{pred}$ values where $m_2 = |E_2|$, $n_3 = |C|$, 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 $O(n_2^2)$ relaxation steps respectively where $n_2 = |P|$ 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 $O(m_2 R + n_3 + k R (m_2 + n_2^2) + n_2 R) = O(k R (m_2 + n_2^2))$. Thus the overall time complexity of the GREEDY-MAX-SPA algorithm is $O(n_2^2 R (m_2 + n_2^2))$. We conclude with the following theorem.

**Theorem 4** Given a SPA instance $I$, a greedy maximum matching in $I$ can be obtained in $O(n_2^2 R (m_2 + n_2^2))$ 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:

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

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 $(l(p_j), v_i)$ 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) In line 3 we replace the negative infinity profile $B^-_R$ with a positive infinity profile $B^+_R$ when initialising $\rho(p_j)$ for every $p_j \in \mathcal{P}$. (iii) Finally we replace both max functions (in lines 3 and 5) 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 $O(n^2R(m_2 + n_2))$ 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 $l_k$, $|M(l_k)| \geq d^-_k(I)$. Given a SPA-L instance, 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 $(l_k, v_i) \in E_4$ are saturated), then $I$ admits no constrained matching. Otherwise we augment $f'$ in $N(I)$ by calling GREEDY-MAX-SPA on $I$, changing line 4 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 \( O(n^2 R(m + n^2)) \) time.

Proof Let \( I \) be a SPA-L instance. Also let \( I' \) be a SPA instance constructed from \( I \) by setting \( d^-(I') = 0 \) and \( d^+(I') = d^-(I) \) for each lecturer \( l_k \). Also let \( M' \) be a greedy maximum matching in \( I' \). If \( f' = f(M') \) is a saturating flow in \( N(I') \). It follows that \( M' \) is a greedy \( k \)-constrained matching in \( I \) where \( k = |M'| \). Thus we may obtain a greedy \((k + 1)\)-constrained matching by calling the GET-MAX-AUG algorithm and passing \( f' \) and \( N(I) \) as parameters. We observe that augmenting path returned by the GET-MAX-AUG algorithm does not cause the number of students assigned to any lecturer to decrease. This condition is sufficient to ensure that Lemmas 1, 2 and 3 hold. Thus we may continue to find augmenting paths in this manner until no such path can be found. The resulting matching is a greedy maximum constrained matching in \( I \).

On the other hand suppose \( f' = f(M') \) is not a saturating flow in \( N(I') \). It is easy to conclude that no constrained matching can exist in \( I \). \( \square \)

Acknowledgments

We would like to thank Ulrich Kraehmer and Mante Zelvyte for valuable discussions concerning the problem studied in this paper.

References

[1] D.J. Abraham. Algorithmics of two-sided matching problems. Master’s thesis, University of Glasgow, Department of Computing Science, 2003.
[2] D.J. Abraham, R.W. Irving, and D.F. Manlove. Two algorithms for the Student-Project allocation problem. Journal of Discrete Algorithms, 5(1):79–91, 2007.
[3] A.H. Abu El-Atta and M.I. Moussa. Student project allocation with preference lists over (student,project) pairs. In Proceedings of ICCEE 09, pages 375–379. IEEE, 2009.
[4] L.R. Ford and D.R. Fulkerson. Flows in Networks. Princeton University Press, 1962.
[5] D. Gusfield and R.W. Irving. The Stable Marriage Problem: Structure and Algorithms. MIT Press, 1989.
[6] C.-C. Huang, T. Kavitha, K. Mehlhorn, and D. Michail. Fair matchings and related problems. In IARCS (FSTTCS 2013), volume 24, pages 339–350. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.
[7] A. Hylland and R. Zeckhauser. The efficient allocation of individuals to positions. Journal of Political Economy, 87(2):293–314, 1979.
[8] R.W. Irving. Greedy matchings. Technical Report TR-2003-136, University of Glasgow, Department of Computing Science, 2003.
[9] R.W. Irving. Greedy and generous matchings via a variant of the Bellman-Ford algorithm. Unpublished manuscript, 2006.
[10] R.W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Rank-maximal matchings. ACM Transactions on Algorithms, 2(4):602–610, 2006.
[11] K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation bounds for the student-project allocation problem with preferences over projects. \textit{Journal of Discrete Algorithms}, 13:59-66, 2012.

[12] D.F. Manlove. \textit{Algorithmics of Matching Under Preferences}. World Scientific, 2013.

[13] D.F. Manlove and G. O’Malley. Student project allocation with preferences over projects. \textit{Journal of Discrete Algorithms}, 6:553–560, 2008.

[14] K. Mehlhorn and D. Michail. Network problems with non-polynomial weights and applications. Unpublished manuscript, 2006.

[15] C.T.S. Sng. \textit{Efficient Algorithms for Bipartite Matching Problems with Preferences}. PhD thesis, University of Glasgow, Department of Computing Science, 2008.

[16] M. Zelvyte. The Student-Project Allocation problem: a network flow model. Honours project dissertation, University of Glasgow, School of Mathematics, 2014.

[17] L. Zhou. On a conjecture by Gale about one-sided matching problems. \textit{Journal of Economic Theory}, 52(1):123–135, 1990.