Selecting PhD Students and Projects with Limited Funding

Jatin Jindal, Jérôme Lang, Katarína Cechlárová, Julien Lesca

Selecting PhD Students and Projects with Limited Funding

Jatin Jindal
Google
India
jatinjindal369@gmail.com

Jérôme Lang
CNRS, PSL
Paris, France
lang@lamsade.dauphine.fr

Katarína Cechlárová
Pavol Jozef Šafárik University
Košice, Slovakia
katarina.cechlarova@upjs.sk

Julien Lesca
Huawei Technologies
Paris, France
julien.lesca@huawei.com

ABSTRACT

In some universities, there is a fixed number of PhD grants, but a larger number of eligible projects and students, each student being allowed to apply on several projects, and a committee builds up a ranking (masterlist) over student-project pairs. The paper analyses three mechanisms to choose the pairs to be funded. The first one, used in some universities, is a greedy mechanism that gives a huge priority to the masterlist and very little to student's preferences. At the other extremity of the spectrum, we have a priority-list variant of student-oriented Gale-Shapley. It is strategyproof and optimal for students, but has one drawback: it pays too little attention to the masterlist, and thus to the committee. Inbetween, we have an intermediate mechanism which can be seen as a good trade-off. Among the properties we study, one is specific to our setting: dynamic monotonicity, which deals with cases when a student suddenly leaves the system in the middle of the process.

ACM Reference Format:
Jatin Jindal, Jérôme Lang, Katarína Cechlárová, and Julien Lesca. 2022. Selecting PhD Students and Projects with Limited Funding. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

We study a student-project allocation problem motivated by the real procedure used at the computer science department of University Paris-Dauphine. The institute (school, university) publishes a set of PhD projects. Each student may apply for several projects. The total number k of available grants is fixed. After evaluating the students' academic records and their presentations, the university creates a master-list of student-project pairs. The motivation for the master list is to obtain a kind of consensus in the university, taking into account several different considerations, some being specific to students (such as academic achievements), some specific to projects (priority to some topics, balance between research groups) and some specific to student-project pairs (fitness of a student for a particular project). Given this master list, and (possibly) students' preferences over projects, we must both *select* a subset of students and a subset of projects to be funded and to *assign* students to projects. Equivalently, we have to select a fixed number of disjoint pairs consisting each of a student and a project.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

1.1 Our contribution

We define several desirable properties for the procedures and for the matchings they output. Then we define several mechanisms for constructing an allocation of students to projects.

The current procedure used in several French universities is what we call in this paper *greedy*. On the one hand, it leads to a lexicographically optimal matching from the point of view of the committee. On the other hand, it has several drawbacks, the first of them being that it neglects students' preferences. Still, given that it is in use in various places, it is worth studying it in detail.

We offer two alternative approaches. They can be seen as variants on the Gale-Shapley Deferred Acceptance algorithms (school-proposing and student-proposing), where the university, via its committee, expresses its preferences by the master list.

We show that the master-list variant of student-oriented Gale-Shapley is a special case of matching with regional quotas, with only one region and its upper quota. It is strategyproof and optimal for students, but has one drawback: it pays too little attention to the masterlist, and thus to the committee. The third mechanism, which we call L-Deferred Acceptance, is intermediate between the other two. Seen from the committee, its outcome is better than student-oriented Gale-Shapley with a master list, and dominated by the greedy mechanism. Seen from students, it is of course not as good as the student-oriented mechanism; and it is incomparable with the greedy mechanism. When the master list is consistent with students' preferences, all three mechanisms are equivalent.

We study the properties of the three mechanisms. We first consider stability and show that all three mechanisms enjoy it. Then we consider manipulation by students: the greedy mechanism can be manipulated by students by declaring some projects unacceptable, the university proposing mechanism can be manipulated by students by stating preferences that are different from their real ones and also by truncating their preference lists, and the student-proposing mechanism is fully incentive compatible.

Finally, we explore a dynamic monotonicity property, which is specific to our setting, and thus novel. In real life, the system is not closed: applicants can also apply to other universities or companies outside the matching mechanism, and thus leave the system at any time. We say that a mechanism satisfies *dynamic monotonicity* if after some student declines an offer, we never have to retract an offer previously made to another student (unless assigning her to a better project). Unlike the other two, the greedy mechanism fails to satisfy it; we suggest a simple way of coping with this failure.

1.2 Related work

The student-project allocation problem has been introduced by Abraham *et al.* [1]. In this model, each project is lead by one lecturer, and both projects and lecturers have capacity constraints. Students have preferences over projects and lecturers have preferences over students. The authors propose a linear-time algorithm to find the student-optimal stable matching. The proposed algorithm is a generalization of the famous Gale-Shapley Deferred Acceptance algorithm [5] where students make proposals to projects. Abraham *et al.* [2] complemented the previous work with a version of the lecturer-oriented Deferred Acceptance algorithm. The authors show that in this algorithm each student who is not unassigned is assigned to the worst project she can have in any stable matching. Further, they proved several structural properties of stable matchings: each lecturer has the same number of students in all stable matchings, exactly the same students are unassigned in all stable matchings and a project offered by an under-subscribed lecturer has the same number of students in all stable matchings.

Manlove and O'Malley [11] consider a model with capacities of projects and lecturers where both students and lecturers have preferences over projects. Stable matchings can have different sizes. The authors prove that finding a stable matching with maximum cardinality is APX-hard, and give an approximation algorithm with a performance guarantee of 2. Iwama *et al.* [8] derive for this problem an improved upper bound of 1.5 and a lower bound of 21/19.

Kwanashie *at al.* [9] use a different optimality criterion for matching students to projects. The preferences of lecturers are ignored and the authors consider the so-called profile, which is a vector whose rth component indicates how many students have their rth-choice project. An efficient algorithm for finding a maximum matching whose profile is lexicographically maximum and another one to find a maximum matching whose reverse profile is lexicographically minimum are proposed.

Abraham *et al.* [1] consider a model where lecturers have preferences over student-project pairs, with a notion of stability as in [3] (see also [10], pages 271-272); the authors extend the student-oriented Deferred Acceptance algorithm to this setting to obtain a stable matching. Our model differs from the above in that we assume a master list of student-project pairs, and consider several different criteria for the obtained matching.

Matching problems where preferences of one or both sides of the market (students and/or schools) are derived from a common master list have been considered by Irving *et al.* [7]. Note that the master list orders agents on only one side of the market; it does not order pairs. Also, in [7] there is no upper bound on the size of the matching. The authors show that in case of strict preferences, there is a unique stable matching that can be obtained by a version of the greedy algorithm.

Cechlárová *et al.* [4] study a problem closely related to ours, however, the focus is on the uncertainty on student decisions to accept or reject a project offered to her. They provide an analysis of the best approximation ratio achievable by a mechanism of exchanges between the committee and the students.

Goto *et al.* [6] study a general model of matching with regions that have minimal and maximal quotas. Their input contains students' preferences over schools, schools' preferences over students, and a master list L over pairs. They propose a generalisation of student-proposing Deferred Acceptance which, under some conditions on the regions, outputs a stable matching and is student-strategyproof. In the special case with a single region and a maximum quota equal to the number of grants, we find something closely related to one of our mechanisms, which will be further discussed in Subsection 4.3.

2 NOTATION

Let $S = \{s_1, \ldots, s_n\}$ be a set of n students and $P = \{p_1, \ldots, p_m\}$ a set of m projects offered by the university. We assume the university has resources to support at most k PhD students. We also assume that no student can work on more than one project and that no project can be assigned to more than one student. Each student s applies to a subset $A(s)$ of projects. Let $\mathcal{A} = \{(s,p) | s \in S; p \in A(s)\}$; a pair (s,p) in $\mathcal{A}$ is called an admissible pair.

Each student s has a preference relation $\succ_s$ over $A(s)$, which we assume to be a strict linear ordering. We write $p_j \succ_s p_r$ if s prefers project p_j to project p_r and $p_j \succeq_s p_r$ means that either $p_j \succ_s p_r$ or $p_j = p_r$. We let $\succ_S = (\succ_{s_1}, \ldots, \succ_{s_n})$. We assume that s prefers a project in $A(s)$ to not getting any project.

The committee (jury) of the university expresses a preference relation over student-project pairs, namely a linear ordering over a subset of $\mathcal{A}$, which we call the *master list* and denote by L. In what follows, we shall use upper indices to denote the ordering of pairs in L; for example, (s^1, p^1) is the first (top-most) pair in L, (s^2, p^2) is the second, etc. We say that (s,p) is preferred to (s',p') by L, which we write $(s,p) \succ_L (s',p')$, if either (1) both (s,p) and (s',p') appear in L, that is, if $(s,p) = (s^i, p^i)$ and $(s',p') = (s^j, p^j)$, and $i < j$, or (2) (s,p) appears in L and (s',p') does not. We will use $(s,p) \succeq_L (s',p')$ to mean that either $(s,p) \succ_L (s',p')$ or $(s,p) = (s',p')$. Note that a student s can appear several times on the list, paired with different projects, and similarly, a project can appear several times on the list, paired with different students.

An instance of the student-project allocation problem with limited funding (SPALF) is a tuple $\mathcal{I} = (S, P, \mathcal{A}, \succ_S, L, k)$.

A matching M is a set of student-project pairs such that no student (respectively, no project) appears in more than one pair in M. A matching M is admissible if $M \subseteq \mathcal{A}$, that is, if no student is assigned to a project she does not apply to. If M is a matching then we denote by $M(s)$ the project to which student s is matched in M and by $M(p)$ the student matched to project p in M. If student s (or project p) is not matched in M then $M(s)$ (or $M(p)$) will be the empty set. By $S(M)$ and $P(M)$ we denote the set of students and projects matched by M, respectively. Formally, $S(M) = \{s \in S | (s,p) \in M \text{ for some } p \in P\}$ and $P(M) = \{p \in P | (s,p) \in M \text{ for some } s \in S\}$.

The university is considered as a special agent whose preferences over matchings are derived lexicographically from the master list L: the university's (or equivalently, the committee's) preference relation over matchings, denoted by $\succ_L^*$, is defined as follows: $M \succ_L^* M'$ if there exists a pair $(s,p) \in \mathcal{A}$ such that $(s,p) \in M \setminus M'$, and for each $(s',p') \succ_L (s,p)$ we have $(s',p') \in M$ if and only if $(s',p') \in M'$. [1]

[1] The rationale for using a lexicographic extension is mainly fairness and accountability to different components of the university (supervisors and research groups): once the committee has made its mind about $\succ_L$, it would be difficult to argue that we must sacrifice a pair against less prioritary pairs; for instance, if $(s_1, p_1) \succ_L (s_1, p_3) \succ_L$

In the real world, finding a final matching may be complicated because of several difficulties; the main two are discussed below.

The first difficulty may be caused by the ordering of the master list L that does not respect the preferences of students: it can be the case that student s prefers p to p' and yet the committee decides to rank (s, p') before (s, p), or even not to rank (s, p) at all in the masterlist. There are various reasons for that: s may be considered to be a better fit for p' than for p though she prefers p; or the supervisor associated with p may have some negative opinion about s.

This difficulty can be avoided by requiring that L is *student-consistent*: whenever $(s, p) \succ_L (s, p')$ then s prefers p to p'. We will see that assuming student-consistency leads to great simplification.

The second difficulty stems from the fact that a student's willingness to commit to a project may evolve: if a student applies for a position elsewhere (*e.g.*, another university, or a company), at some time point she may decline on offer that she has already accepted, and the current matching must be updated.

In what follows, we denote by $L^{(s,p)}$ and $M^{(s,p)}$ the set of (student-project) pairs in L and M respectively, that are strictly preferred by L to the pair (s, p). Formally, $L^{(s,p)} = \{(s', p') \in L | (s', p') \succ_L (s, p)\}$ and $M^{(s,p)} = \{(s', p') \in M | (s', p') \succ_L (s, p)\}$. We will use the notation M_{least} to denote the least preferred pair in the matching M according to the master list L. Formally $M_{least} = \{(s, p) \in M | \forall (s', p') \in M, (s', p') \succeq_L (s, p)\}$.

Finally, we denote by $\mathcal{M}$ a mechanism, which takes in input an instance $\mathcal{I}$ and returns a matching $M \subseteq S \times P$, *i.e.*, $\mathcal{M}(\mathcal{I}) = M$.

Example 1. *Assume we have four students $S = \{s_1, s_2, s_3, s_4\}$, four projects $P = \{p_1, p_2, p_3, p_4\}$ and three grants $k = 3$. We take $A(s_1) = A(s_2) = P$, $A(s_3) = \{p_1, p_3, p_4\}$, $A(s_4) = \{p_1, p_2, p_4\}$. The students' preferences and the master list are as below.*

Students' preferences		*Master list L (continued over 3 lines)*
$\succ_{s_1}$:	p_2, p_3, p_1, p_4	$(s_1, p_1), (s_2, p_1), (s_1, p_3), (s_2, p_3),$
$\succ_{s_2}$:	p_1, p_2, p_4, p_3	$(s_3, p_3), (s_4, p_1), (s_2, p_4), (s_1, p_2),$
$\succ_{s_3}$:	p_1, p_4, p_3	$(s_2, p_2), (s_3, p_4), (s_3, p_1), (s_4, p_2), (s_4, p_4)$
$\succ_{s_4}$:	p_4, p_2, p_1	

Students' preferences	$\succ_{s_1}$:	p_2, p_3, p_1, p_4
	$\succ_{s_2}$:	p_1, p_2, p_4, p_3
	$\succ_{s_3}$:	p_1, p_4, p_3
	$\succ_{s_4}$:	p_4, p_2, p_1
Master List	L :	$(s_1, p_1), (s_2, p_1), (s_1, p_3), (s_2, p_3), (s_3, p_3),$ $(s_4, p_1), (s_2, p_4), (s_1, p_2), (s_2, p_2), (s_3, p_4),$ $(s_3, p_1), s_4, p_2), (s_4, p_4)$

We have $(s^1, p^1) = (s_1, p_1)$, $(s^2, p^2) = (s_2, p_1)$ etc. $\succ_L$ is not student-consistent: $(s_1, p_1) \succ_L (s_1, p_2)$ although s_1 prefers p_2 to p_1. Note that p_4 is deemed admissible by s_1 but (s_1, p_4) is not deemed admissible by the committee, as it is not in L.

(s_2, p_1), then preferring larger matchings, i.e., $\{(s_1, p_3), (s_2, p_1)\} \succ \{(s_1, p_1)\}$, would be hard to justify to the research group of the supervisors of p_1.

3 CRITERIA

3.1 Lexicographic Optimality

An admissible matching M is *lexicographically optimal* if no other admissible matching is lexicographically preferred to M, that is, if $M' \succ_L^* M$ holds for no feasible matching M'. Since $\succ_L^*$ is a strict linear order, there is a unique lexicographically optimal matching. In Example 1, the lexicographically optimal matching is $M^* = \{(s_1, p_1), (s_2, p_3), (s_3, p_4)\}$.

3.2 Stability

Definition 1. *Let $M \star (s, p)$ be the matching obtained from M by the following operations:*

(1) *match s and p*
(2) *unmatch s if s was matched in M*
(3) *unmatch p if p was matched in M*
(4) *unmatch M_{least} if $s \notin S(M)$, $p \notin P(M)$, and $|M| = k$.*

Then $(s, p) \in L \setminus M$ is a blocking pair if

(i) $s \in S(M)$, $p \succ_s M(s)$, and $M \star (s, p) \succ_L^* M$.
(ii) $s \notin S(M)$ and $M \star (s, p) \succ_L^* M$

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

This is a rather classical stability condition, seeing the university as an agent whose preferences are lexicographic. Note that matching s to p when (s, p) is blocking may lead to a matching with smaller cardinality. For the preferences given in Example 1 and matching $M = \{(s_2, p_2), (s_3, p_1), (s_4, p_4)\}$:

- (s_2, p_1) is a blocking pair because of condition (i).
- (s_1, p_1) is a blocking pair because of condition (ii).
- (s_1, p_3) is a blocking pair because of condition (ii), since $M_{least} = (s_4, p_4)$.

By contrast, matching $M' = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$ admits no blocking pair and so it is stable.

3.3 Strategyproofness

We shall consider two different kinds of manipulations. To be able to define the needed notions, we first introduce some notation.

Given a student s, we denote by $\succ_{-s}$ the preferences of all students except s. Similarly, by $\mathcal{A}_{-s}$ we denote the admissible pairs of all students except s. Then by $\mathcal{I} + (\succ_s', \succ_{-s})$ we denote the instance $\mathcal{I}$ modified in such a way that the preference relation $\succ_s$ of student s was changed to a preference relation $\succ_s'$, while everything else, including $A(s)$ was kept intact. Notation $\mathcal{I} + (A'(s), \mathcal{A}_{-s})$ means that the instance $\mathcal{I}$ is modified in such a way that student s replaced her admissible set $A(s)$ by its strict subset $A'(s)$; the preference ordering of s on $A'(s)$ is simply the restriction of $\succ_s$ to $A'(s)$, and L is simplified accordingly.

We say that mechanism $\mathcal{M}$ is *permutation strategyproof* if no student can force a better outcome for her by permuting her preference list; formally, if for all $s \in S$ we have $\mathcal{M}(\mathcal{I}) \succeq_s \mathcal{M}(\mathcal{I} + (\succ_s', \succ_{-s}))$ for all possible $\succ_s'$. We say that mechanism $\mathcal{M}$ is *truncation strategyproof* if no student can force a better outcome for her by truncating her preference list; formally, if for all $s \in S$ we have $\mathcal{M}(\mathcal{I}) \succeq_s \mathcal{M}(\mathcal{I} + (A'(s), \mathcal{A}_{-s}))$ for all possible $A'(s) \subseteq A(s)$.

3.4 Dynamic monotonicity

So far we assumed that the final matching is computed at once, from the master list and the students' preferences. In real life, things are sometimes not so static: some students may restrict further their set of admissible projects, or even withdraw completely. To take an example, in the university of two of the authors, the master list is made up in May, but the contracts are signed in July. Between these dates, a student may get a better offer *from another place*. This means that s may accept p at time t (say, May 15), and then, on June 1, decline the offer (and then either leave the system completely, or wait until she possibly gets a project she prefers to the project she got elsewhere – and *a fortiori*, that she prefers to p). In other terms, a student maintains a threshold above which she is ready to accept a project, and this threshold may increase with time.

This calls for adapting our mechanisms so as to *update* the current matching as soon as the student rejects a project p she is currently assigned to (and implicitly, all projects below p in her preference relation). However, updating a matching can be troublesome, as the following example shows.

Example 2.

Students' preferences	*Master list L*
$\succ_{s_1}$: p_2, p_1	$(s_1, p_1), (s_1, p_2), (s_2, p_2), (s_3, p_1)$
$\succ_{s_2}$: p_2	
$\succ_{s_3}$: p_1	

Assume that our mechanism outputs the lexicographically optimal matching $M^* = \{(s_1, p_1), (s_2, p_2)\}$. *However, if a few days after the proposal,* s_1 *ends up rejecting the offer* p_1 *by restricting her admissible set to* $\{p_2\}$, *then the new lexicographically optimal matching is* $\{(s_1, p_2), (s_3, p_1)\}$. *Thus, the following dilemma appears:*

- *we can make* $\{(s_1, p_2), (s_3, p_1)\}$ *the new output. Lexicographic optimality is preserved, but this is very rude to* s_2, *to whom we made an offer, who perhaps has already accepted this offer, and who is now told that the offer is cancelled, and even worse, that the new matching does not match her to any project.*
- *keep the offer* (s_2, p_2) *and compute the lexicographically optimal matching containing this pair, namely,* $\{(s_2, p_2), (s_3, p_1)\}$. *But we lose lexicographic optimality, as this matching is not lexicographically optimal for the updated profile.*

The reason why evolving preferences may either lead to a suboptimal matching or to canceling a previous offer (and possibly end up giving the concerned student a worse final outcome than this offer) is due to the failure of *dynamic nonmonotonicity*, which we will define shortly.

Definition 2. *Let* $\mathcal{I} = (S, P, \mathcal{A}, \succ_S, L, k)$ *be an instance of SPALF,* $s_i \in S$, *and* $p^* \in A(s_i)$. *We define* $\mathcal{I}[s_i, p^*] = (S, P, \mathcal{A}', \succ'_S, L, k)$, *where* $A'(s_i) = \{p, p \succ_{s_i} p^*\}$, $\succ'_{s_i}$ *is the restriction of* $\succ_{s_i}$ *to* $A'(s_i)$, *and for all* $j \neq i$, $A'(s_j) = A(s_j)$ *and* $\succ'_{s_j} = \succ_{s_j}$.

Replacing $A(s_i)$ by $A'(s_i) = \{p, p \succ_s p^*\}$ corresponds to student s_i rejecting all proposals that she does not prefer to p^*. Since $p^* \in A(s_i)$, we have $A'(s_i) \subset A(s_i)$. A particular case is obtained when p^* is the student's most preferred project: in this case, $A' = \emptyset$, which means that s will not accept any project from that point, and thus withdraws from the system.

Definition 3. *Let* $\mathcal{M}$ *be a SPALF mechanism and* $\mathcal{I}$ *a SPALF instance. A proposal* (s, p) *is* safe *for* $\mathcal{I}$ *and* $\mathcal{M}$ *if for any instance* $\mathcal{I}[s_i, p^*]$, *if* $M = \mathcal{M}(\mathcal{I})$ *and* $M' = \mathcal{M}(\mathcal{I}[s_i, p^*])$, *if* $M(s) \in A'(s)$ *then* $M'(s) \succeq_s M(s)$. *A mechanism* $\mathcal{M}$ *satisfies* dynamic monotonicity *if for any instance* $\mathcal{I}$, *every pair in* $\mathcal{M}(\mathcal{I})$ *is safe.*

When a proposal (s, p) is not safe then the following may happen: p has been proposed to s and s has accepted, and at some later point, the mechanism has to come back on this proposal and to assign s to a project that she likes less than p, or even to no project at all. Dynamic monotonicity guarantees that this will never occur.

4 MECHANISMS

4.1 Greedy mechanism

This mechanism, called *Greedy* for short, goes down the master list L making proposals to the next (student-project) pair (s, p) such that both s and p are currently unmatched. This is repeated until the current matching has reached the maximal capacity k or until the master list L is exhausted. Algorithm 1 gives a formal description.

Algorithm 1: Greedy Mechanism

```
Input: Instance I = (S, P, A, ≻_S, L, k)
1 M := ∅
2 i := 1
3 while |M| < k and i ≤ |L| do
4     if s^i ∉ S(M) and p^i ∉ P(M) then
5         M := M ∪ {(s^i, p^i)}
6     i := i + 1
7 return M
```

In Example 1, the output is $M_G = \{(s_1, p_1), (s_2, p_3), (s_3, p_4)\}$.

Example 3. $S = \{s_1, s_2\}$, $P = \{p_1, p_2, p_3\}$ *and* $k = 2$. *We assume that* $A(s_1) = \{p_1, p_2\}$, $A(s_2) = \{p_3\}$. *The students' preferences and the master list are given in the following table.*

Students' preferences	*Master list L*
$\succ_{s_1}$: p_2, p_1	$(s_1, p_1), (s_1, p_2), (s_2, p_3)$
$\succ_{s_2}$: p_3	

Greedy *returns the matching* $\{(s_1, p_1), (s_2, p_3)\}$.

Theorem 1. *Greedy outputs a lexicographically optimal matching.*

Proof. Let M be the matching given by *Greedy* and let M' be a matching that is lexicographically preferred to M. Let (s, p) be the pair in $M' \setminus M$ with highest priority according to L. Such a pair exists since $M' \succ^*_L M$. Then, during the execution of *Greedy*, there was a moment when pair (s, p) was considered in line 4, otherwise $M_{least} \succ_L (s, p)$, implying $M \succ^*_L M'$. The fact that $(s, p) \notin M$ means (see the condition in line 4 of *Greedy*) that either there is a pair $(s, p') \in M$ such that $(s, p') \succ_L (s, p)$ or there is a pair $(s', p) \in M$ such that $(s', p) \succ_L (s, p)$. The choice of (s, p) implies that $(s, p') \in M'$ or $(s', p) \in M'$, respectively, and so, by the definition of a matching, $(s, p) \notin M'$, a contradiction. □

Theorem 2. *The matching returned by Greedy is stable.*

Proof. Suppose that the matching M returned by Algorithm 1 is not stable, that is, there exists a blocking pair (s, p) for M.

If $M_{least} \succ_L (s, p)$, then (s, p) cannot be a blocking pair. Hence $(s, p) \succ_L M_{least}$, and we must have reached line 5 of the algorithm for pair (s, p) and the condition must have failed. Hence, either $(M(s), p) \succ_L (s, p)$ or $(s, M(s)) \succ_L (s, p)$, in either case (s, p) cannot be a blocking pair. □

Theorem 3. *Greedy is permutation strategyproof, but not truncation strategyproof.*

Proof. *Greedy* does not take into account the preferences of students. Hence, no student has an incentive to reveal different preferences since that will not affect the matching selected.

Example 3 provides a case where a student can get a better outcome by revealing a different set of acceptable projects. If s_1 reports $A'(s_1) = \{p_2\}$ instead of her true admissible set then she will be assigned to p_2 instead of her less preferred project p_1. □

Note that *Greedy* heavily relies on the master list and does not take students' preferences into account[2], therefore it is meant to be used if we want to give priority to the opinion of the committee over those of the students.

Theorem 4. *Greedy does not satisfy dynamic monotonicity.*

Proof. See Example 2. □

Since *Greedy* is not monotonic, finding safe offers is nontrivial. Still, a maximal set of safe offers can be found in polynomial time:

Proposition 1. *Let $\mathcal{I} = (S, P, \mathcal{A}, \succ_S, L, k)$ be an instance of SPALF. An offer (s, p) is safe if the following two conditions hold:*

1. *there is no pair (s', p') such that $(s', p') \succ_L (s, p)$ and either $s = s'$ or $p = p'$.*
2. *the maximum cardinality matching in $M^{(s,p)}$ has cardinality at most $k - 1$.*

Proof. Assume (1) and (2) hold. The algorithm must consider the pair (s, p): if it does not, then it means it stops before with a matching of cardinality k; but then $M^{(s,p)}$ would have cardinality k. Once it considers it, because of condition 1, at this step neither $s \in S(M)$ nor $p \in S(M)$, therefore the output will contain (s, p). □

To compute a maximal set of safe offers, it suffices to determine, for each student s, if there is a safe offer for s. On Example 2, (s_1, p_1) is a safe offer, but there is no safe offer for s_2 or s_3.

Proposition 1 provides only a sufficient condition for an offer to be safe. The following example, with $k = 2$, shows that it is not necessary.

Students' preferences	Master list L
$\succ_{s_3}$: p_1, p_3	$(s_1, p_1), (s_3, p_1), (s_3, p_3)$

Offer (s_1, p_3) is safe but does not satisfy the conditions of Proposition 1.

[2]More precisely, it takes into account the set of projects they declare admissible, but not their relative preferences between projects.

4.2 Masterlist Proposing Deferred Acceptance

For brevity, we shall denote this mechanism by LDA (as L-Deferred Acceptance), as the proposals are made according to the master list L. The mechanism goes down the master list L making proposals to the next (student-project) pair (s, p) such that project p is not matched in the current matching, provided the funding capacity is not yet exhausted. After acceptance, the pair is added to the current matching. The student may accept the proposal either if she is not matched or if she prefers project p to her current assignment $M(s)$. If the latter occurs, pair $(s, M(s))$ is deleted from M (otherwise M is left unchanged) and the proposal sequence starts immediately after $(s, M(s))$ according to L. This is repeated while the size of the current matching is smaller than the funding capacity k.

Algorithm 2: LDA Mechanism

```
   Input: Instance I = (S, P, A, ≻_S, L, k)
1  M := ∅
2  i := 1
3  while |M| < k and i ≤ |L| do
4      (s, p) := (s^i, p^i)
5      i := i + 1
6      if p ∉ P(M) then
7          if s ∉ S(M) then
8              M := M ∪ {(s, p)}
9          else
10             if p ≻_s M(s) then
11                 M := M\{(s, M(s))} ∪ {(s, p)}
12                 Let t be such that (s, M(s)) = (s^t, p^t)
13                 i := t + 1
14 return M
```

For the instance in Example 1, LDA first considers (s_1, p_1) and then (s_1, p_3). As s_1 prefers p_3 to p_1, the pair (s_1, p_1) is rejected and we next consider the pair immediately following the pair that has just been deleted, namely to (s_2, p_1). This proposal is accepted and at this point $M = \{(s_1, p_3), (s_2, p_1)\}$. The mechanism continues with (s_2, p_4), which is rejected. Then (s_1, p_2), which is accepted and thus (s_1, p_3) is rejected. Now $M = \{(s_1, p_2), (s_2, p_1)\}$ and the next pair considered is (s_2, p_3), which is rejected, and then (s_3, p_3), which is accepted. The final matching is $M_{LDA} = \{(s_1, p_2), (s_2, p_1), (s_3, p_3)\}$.

Observation 1. *At each point of the execution of Algorithm 2, each student is either unmatched or matched to a project that she weakly prefers to any project assigned to her during previous steps.*

We say that a pair (s, p) is *considered* by Algorithm 2 (on a specific instance), if at some point during the execution, the current pair in the algorithm is (s, p). We say that (s, p) is *proposed (or offered)* if at some point of the execution, the current pair is (s, p) and line 7 of the algorithm is reached.

Since LDA goes down the master list, possibly revisiting multiple times the same pair, the following obviously holds:

Observation 2. *At the step where (s,p) is considered by Algorithm 2, each pair (s',p') such that $(s',p') \succ_L (s,p)$ has already been considered earlier during the algorithm.*

The following lemma expresses a property of proposals made by the LDA Mechanism.

Lemma 1. *Suppose (s,p) is considered at some step of Algorithm 2 and let M be the current matching obtained at the end of the previous step. Then for every student $s' \in S$ and for every $p' \in P \setminus P(M)$, $(s',p') \succ_L (s,p)$ implies $M(s') \succeq_{s'} p'$.*

Proof. By contradiction. Assume that (s,p) is considered at some step σ, that M is the matching at the end of the previous step $\sigma - 1$, and that (1) $(s',p') \succ_L (s,p)$, (2) $p' \notin P(M)$, and (3) $p' \succ_{s'} M(s')$. From (1) and Observation 2, (s',p') must have been considered before (s,p). If (s',p') was proposed, then by Observation 1, $M(s') \succeq_{s'} p'$, contradicting $p' \succ_{s'} M(s')$. Therefore, (s',p') was never proposed. Since at step σ we have $p' \succ_{s'} M(s')$, this means that (4) whenever (s',p') is considered earlier at step $\sigma' < \sigma$ with matching $M_{\sigma'}$, we have $p' \in P(M_{\sigma'})$ (see condition on line 5).

Now we claim that (5) for all $\sigma' < \sigma$ such that $p' \in P(M_{\sigma'})$, we have $(M_{\sigma'}(p'),p') \succ_L (s',p')$. We will prove this by induction. For the base case, let σ_1 be the first step (s',p') was considered. By (4), $p' \in P(M_{\sigma_1})$, and since no pair below (s',p') is considered at step σ_1, we have $(M_{\sigma_1}(p'),p') \succ_L (s',p')$. Our induction hypothesis is that if at some step σ' when we considered (s',p') pair we have $(M_{\sigma'}(p'),p') \succ_L (s',p')$, the next step σ'' when we consider (s',p'), we will have $(M_{\sigma''}(p'),p') \succ_L (s',p')$. To prove this hypothesis, we can see that when p' is unassigned for the first step after σ', the current pair (s^i,p^i) would be next pair after $(M_{\sigma'}(p'),p')$ in the master list (corresponding to lines 12-13 of Algorithm 2). So $(s^i,p^i) \succeq_L (s',p')$, p' is unassigned now, but by (4) when we considered pair (s',p'), p' is assigned. This is only possible if p' is assigned at some step in between. But until then the current pair would always appear before (s',p') in L. This implies that $(M_{\sigma''}(p'),p') \succ_L (s',p')$ and proves our induction hypothesis and therefore (5).

In the proof of (5), whenever p' is unassigned, the pair under consideration is preferred to (s',p') by L. But after the end of step $\sigma - 1$, by (1) $p' \notin P(M)$, which contradicts the above argument. □

Corollary 4.1. *Let M be the output of LDA, and M' be the matching just before $M_{least} = (s,p)$ is proposed. Then $P(M) \subseteq P(M') \cup \{p\}$.*

Proof. Assume that $P(M) \not\subseteq P(M') \cup \{p\}$: M contains a pair (s',p') such that $p' \notin P(M')$ and $p' \neq p$. $(s',p') \in M$ implies $(s',p') \succeq_L M_{least}$. Now, we claim that $M(s') \succ_{s'} p'$.

From Lemma 1, $p' \notin P(M')$ and $(s',p') \succ_L (s,p)$ imply $M'(s') \succ_{s'} p'$. By Observation 1, we have $M(s') \succeq_{s'} M'(s')$. Hence $M(s') \succ_{s'} p'$, which contradicts $M(s') = p'$. Hence $P(M) \subseteq P(M') \cup p$. □

Theorem 5. *LDA outputs a stable matching.*

Proof. By contradiction, suppose that the matching given by LDA, say M, is not stable, that is, there exists a blocking pair (s,p).

If $p \in P(M)$ then $(s,p) \succ_L (M(p),p)$ should hold for (s,p) to be a blocking pair. Furthermore, p was unassigned during the step right before it was proposed to agent $M(p)$. Therefore, we know by Lemma 1 that $M(s) \succeq_s p$, leading to a contradiction with (s,p) being blocking (conditions 1). Hence $p \notin P(M)$.

Now, s never received the offer (s,p) during the execution of Algorithm 2, since otherwise she would have accepted and by Observation 1, she would have been matched in M to a project that she weakly prefers to p. Therefore, either (i) Algorithm 2 stopped before considering pair (s,p) or (ii) project p was matched to another student each time (s,p) was considered.

In case (i), because (s,p) was never considered, and the last pair to be considered comes before (s,p) in L. Therefore, both $M_{least} \succ_L (s,p)$ and $|M| = k$ hold. This contradicts the assumption that (s,p) is blocking: neither conditions 1 nor condition 2 can hold.

In case (ii), assume that p was matched with s' when (s,p) was considered by Algorithm 2. If $(s,p) \succ_L (s',p)$, then by Lemma 1, at the time when s' is offered to p, s was matched to a project that he prefers to p, which by Observation 1, contradicts $p \succ_s M(p)$. Therefore it must be the case that $(s',p) \succ_L (s,p)$.

Therefore, whenever (s,p) is considered, p is matched to some student s' such that $(s',p) \succ_L (s,p)$. *A fortiori*, whenever an offer (s'',p'') such that $(s,p) \succ_L (s'',p'')$ is made, p is already matched at that stage. Let M' (respectively M'') be the matching just before (respectively after) we proposed the offer $M_{least} = (s''',p''')$. Because (s,p) is blocking, $(s,p) \succ_L M_{least}$ (by conditions 1 and 2). Now, p must be already matched when a pair with smaller priority in L is proposed. Hence $p \in P(M')$. If $p \notin P(M'')$ then $|M''| = |M'| < k$. Otherwise, Algorithm 2 will still execute at least until p become free. This means that in both cases $|M''| < k$ and $|M'| < k$. From Corollary 3, we know that every student will take the offer inside $P(M') \cup \{p'''\}$. Hence, $P(M) \subseteq P(M') \cup p''' \setminus p$. This means that $|P(M)| < k$, which is not possible as the mechanism stopped before proposing the offer (s,p), which can be proposed. Hence LDA returns a stable matching. □

Theorem 6. *The LDA Mechanism is neither permutation nor truncation strategyproof.*

Proof. Let $k = 2$, $L = \{(s_1,p_1),(s_2,p_1),(s_1,p_2),(s_1,p_3)\}$, $A(s_1) = \{p_1,p_2,p_3\}$, $A(s_2) = \{p_1\}$. If $\succ_{s_1} = p_3,p_2,p_1$ then the output is $\{(s_1,p_2),(s_2,p_1)\}$.

Permutation: If s_1 reports $\succ'_{s_1} = p_3,p_1,p_2$, then the output is $\{(s_1,p_3),(s_2,p_1)\}$: s_1 has an incentive to report $\succ'_{s_1}$ instead of $\succ_{s_1}$.

Truncation: If s_1 reports $A'(s_1) = \{p_3\}$ then the output is $\{(s_1,p_3),(s_2,p_1)\}$, thus s_1 has an incentive to report $A'(s_1)$. □

Proposition 2. *LDA satisfies dynamic monotonicity.*

Proof. Assume not: then there exist instances $\mathcal{I}$, $\mathcal{I}' = \mathcal{I}[s_i,p^*]$, such that $LDA(\mathcal{I}) = M$, $LDA(\mathcal{I}') = M'$, and for some s_i, $M(s_i) \succ_{s_i} M'(s_i)$. Let S be the set of all pairs $(s,M'(s))$ such that $M(s) \succ_s M'(s)$.

Assume first that $S \neq \emptyset$. Let (s,p) is the L-most preferred pair of S. We claim that $(s,M(s))$ is a blocking pair in M'. Assume it is not. Then we are in one of these three cases:

(1) $(s,M(s)) \notin L'$: since $(s,M(s)) \in L$, we have $s = s_i$, that is, s is the student who has reduced her admissibility threshold. But then, $M(s_i) \succ_{s_i} M'(s_i)$, we cannot have $(s,p) \in M'$ as $p \notin A'(s))$.

(2) There is a $(s',M(s)) \in M'$ such that $(s',M(s)) \succ_{L'} (s,M(s))$: $(s',M(s))$ is not blocking in M because M is stable, therefore $M(s') \succ_{s'}$

$M(s) = M'(s')$. So, $(s', M(s')) \in S$, and $(s', M(s') = M(s)) \succ_{L'} (s, p)$, which contradicts the assumption that (s, p) is the L-most preferred pair of S.
(3) $M'_{least} \succ'_L (s, M(s))$ and $|M'| = k$: This implies that $M'_{least} \succ'_L M_{least}$ and $M'_{least} \succ_L M_{least}$. Now, the k different pairs in M' are all in L so the algorithm should have selected them all in M too, contradicting $M(s_i) \succ_{s_i} M'(s_i)$.

Now, assume $S = \emptyset$. Since $M(s_i) \succ_{s_i} M'(s_i)$, we must have $s_i \notin S(M')$. Again, assume (s_i, p_i) is not blocking for M'. Then we are in one of these two cases:
(1) $(s', p_i) \succ_{L'} (s_i, p_i)$ for some $(s', p_i) \in M'$. Since (s', p_i) is not weakly blocking in M, $M(s') \succeq_{s'} p_i$, therefore $M(s') \succeq_{s'} M'(s')$. But then $(s', M(s'))$ should be in S, which contradicts $S = \emptyset$.
(2) $M'_{least} \succ'_L (s_i, p_i)$ and $|M'| = k$: This implies that $M'_{least} \succ'_L M_{least}$ and thus $M'_{least} \succ_L M_{least}$. A similar reasoning as above leads to a contradiction with $M(s_i) \succ_{s_i} M'(s_i)$. □

We end up this section by a remark. In Algorithm 2 we could consider replacing line 3 by simply **While** $|M| < k$ **and** $i \leq |L|$, and line 6 by **If** $p \notin P(M)$ **and** $|M| < k$, thus allowing already matched students to be made offers even after maximal capacity is reached, and to switch to the replace their previous match by the new offer. However, if we do that we lose stability, as can be seen on this example: $L = (s1, p1), (s_3p_1), (s_2, p_2), (s_1p_3)$, and, $k = 2$. If s_1 prefers p_3 to p_1 then Algorithm 2 stops at $|M| = 2$ and outputs $\{(s_1, p_1), (s_2, p_2)\}$. But the modified algorithm with the weaker stopping condition outputs $\{(s_1, p_3), (s_3, p_1)\}$. Now, (s_2, p_2) is blocking: it has priority over (s_3, p_1) in L, and s_2 prefers p_2 to nothing.

4.3 Student proposing Gale Shapley

This mechanism, abbreviated by SGS, is very similar to the classical student-proposing Deferred-Acceptance algorithm. In the beginning, all students are unassigned and apply for their most preferred project. A first selection is made according to the masterlist, according to the choice function Ch (see further). All students who are rejected then apply to their second best project, and so on until all students are either assigned to a project, or have exhausted their preference list. The mechanism is formally given as Algorithm 3.

The choice function Ch is reminiscent of *Greedy*, except that it is applied to a set of offers X' containing at most one offer per student. These offers are considered in the decreasing priority order (according to L) until the maximum number of grants is reached.

Given an instance $\mathcal{I} = (S, P, \mathcal{A}, \succ_S, L, k)$, a set of rejected offers $Re \subseteq S \times P$, and a student s, we define $O(s, Re)$ as her most preferred project p for which she has not been rejected, if such a p exists; and $O(s, Re)$ is undefined otherwise. Formally, $O(s, Re) = p$ if $(s, p) \notin Re$, $p \in A(s)$ and for all $p' \succ_s p$ we have $(s, p') \in Re$.

Given a set of offers X' containing at most one offer per student, $Ch(L, X')$ is the selected set of offers defined by Algorithm 3.

Example 4. *We run SGS on the instance of Example 1. Initially, $Re = X = \emptyset$. Then $X' = \{(s_1, p_2), (s_2, p_1), (s_3, p_1), (s_4, p_4)\}$. Reordering X' w.r.t. L gives $o_1 = (s_2, p_1), o_2 = (s_1, p_2), o_3 = (s_3, p_1), o_4 = (s_4, p_4)$, $X = Ch(X', \mathcal{I}) = \{(s_1, p_2), (s_2, p_1), (s_4, p_4)\}$, $Re = \{(s_3, p_1)\}$.*
At the next step we have $X' = \{(s_1, p_2), (s_2, p_1), (s_3, p_4), (s_4, p_4)\}$, $X = \{(s_1, p_2), (s_2, p_1), (s_3, p_4)\}$ and $Re = \{(s_3, p_1), (s_4, p_2)\}$.

Algorithm 3: Function Ch

Input: L, X'
Let $X' := \{o_1, \ldots, o_{|X'|}\}$, such that $o_1 \succ_L \ldots \succ_L o_{|X'|}$
$i := 1$
$X := \emptyset$
while $i \leq |X'|$ *and* $|X| < k$ **do**
  Let $o_i = (s, p)$
  if $p \notin P(X)$ **then**
    $X := X \cup \{o_i\}$
  $i := i + 1$

Algorithm 4: Student proposing Gale Shapley Mechanism

Input: Instance $\mathcal{I} = (S, P, \mathcal{A}, \succ_S, L, k)$
$Re := \emptyset$
$X := \emptyset$
while *some student s is unassigned in X and has not exhausted her admissible set $A(s)$* **do**
  $X' = \{(s, O(s, Re)) : s \in S, O(s, Re)$ is defined $\}$
  $X := Ch(L, X')$
  $Re := Re \cup (X' \setminus X)$
return X

At the next step we have $X' = \{(s_1, p_2), (s_2, p_1), (s_3, p_4), (s_4, p_2)\}$, X does not change, and $Re = \{(s_3, p_1), (s_4, p_4), (s_4, p_2)\}$.
At the next step we have $X' = \{(s_1, p_2), (s_2, p_1), (s_3, p_4), (s_4, p_1)\}$, X does not change, and $Re = \{(s_3, p_1), (s_4, p_4), (s_4, p_2), (s_4, p_1)\}$. s_4 has exhausted her preference list and all other students are assigned, so the returned matching is $\{(s_1, p_2), (s_2, p_1), (s_3, p_4)\}$.

This algorithm is however not new. It corresponds to the Priority list based Deferred Acceptance mechanism with regional minimum and maximum quotas, defined in [6], where (1) projects corresponds to universities; (2) each project has a maximum quota 1, and no minimum quota; and (3) there is only one region, containing all projects, with maximum quota k and no minimum quota.

There is only one assumption in [6] which we do not have: they assume that all universities are acceptable to all students and *vice versa*. However, this assumption is needed only to ensure the existence of a feasible matching in presence of minimum quotas. It is not needed if there are no minimum quotas, and does not play any role in the proofs of the results that we refer to below. This allows us to import several of their results, namely:

Corollary 4.2. *[Theorem 6 in [6]]*
SGS outputs a stable matching.

Corollary 4.3. *[Theorem 5 in [6]]*
SGS is permutation strategyproof.

Corollary 4.4. *[Theorem 9 in [6]]*
SGS is student-optimal among all stable matchings.

Results in [6] do not say anything about truncation strategyproofness, because they assume all schools are acceptable to students.

Proposition 4.5. *SGS is truncation strategyproof.*

Proof. SGS tentatively assigns projects to students in the decreasing preference order. Assume $SGS(\mathcal{I}) = M$ and $M(s) = p$. This means that all offers of a project that s prefers to p have been rejected. Assume now that s truncates her preference list below p', and let M' be the resulting matching. If $p' \succ_s p$, then s will be unsassigned in M'. If $p \succeq_s p'$, then $M' = M$. Therefore, SGS is truncation strategyproof. □

Lemma 2. *Let M^* be the output of SGS on a given instance $\mathcal{I}$, let s be a student and let p be a project in $A(s)$ and M' be the output of SGS on instance $\mathcal{I} + (A'(s), \mathcal{A}_{-s})$. Then for any student $s' \in S$, we have $M'(s') \succeq_{s'} M^*(s')$.*

Proof. Matching M^*, which was stable with respect to $\mathcal{I}$, is still stable with respect to $\mathcal{I} + (A'(s), \mathcal{A}_{-s})$ because any blocking pair in $\mathcal{I} + (A'(s), \mathcal{A}_{-s})$ will be blocking in $\mathcal{I}$ as well. Hence, every student weakly prefers M' over M^* because of Corollary 4.5. □

Proposition 3. *SGS satisfies dynamic monotonicity.*

Proof. Let $M = SGS(\mathcal{I})$ and $M' = SGS(\mathcal{I}')$ where $\mathcal{I}' = \mathcal{I}[s_i, p]$. Repeatedly applying Lemma 2 gives: for all s', $M'(s') \succeq_{s'} M(s')$. □

4.4 Discussion

We first observe that the restriction to student-consistent master lists makes all there mechanisms coincide.

Theorem 7. *If the master list is student consistent, then the matchings given by* Greedy, *LDA and SGS coincide.*

Proof. Let M^G, M^{LDA}, M^{SGS} denote the matchings that are output by Greedy, LDA and SGS, respectively.

Suppose $M^G \neq M^{SGS}$. Since M^G is lexicographically optimal, there exists $(s,p) \in \mathcal{A}$ such that $(s,p) \in M^G \setminus M^{SGS}$ and for any $(s',p') \in L^{(s,p)}$, $(s',p') \in M^G \Leftrightarrow (s',p') \in M^{SGS}$. This implies that $(s,p) \succ_L (s, M^{SGS}(s))$ and $(s,p) \succ_L M^{SGS}_{least}$. Since the master list is student consistent, hence $p \succ_s M^{SGS}(s)$. Therefore, (s,p) is the blocking pair in M^{SGS} which is not possible. Hence, $M^G = M^{SGS}$.

Now we show that $M^G = M^{LDA}$. When applying LDA, once an offer is assigned then a student will never deviate from that project because the master list is student-consistent. So, this basically reduces to going once top-down in the master-list in the similar way as in *Greedy*. Therefore, the output will be same. □

As a consequence, under the assumption that the master list is student-consistent, all the considered mechanisms output the student-optimal stable matching and are permutation as well as truncation strategyproof.

Next, we compare the three mechanisms according to (1) the committee's preference expressed by the master list, and (2) the students' preferences.

Proposition 4. *For any $\mathcal{I}$, $Greedy(\mathcal{I}) \succeq_L LDA(\mathcal{I}) \succeq_L SGS(\mathcal{I})$.*

Proof. $Greedy(\mathcal{I}) \succeq_L LDA(\mathcal{I})$ is a direct consequence of Proposition 1. Now, assume $SGS(\mathcal{I}) \succ_L SGS(\mathcal{I})$ for some instance L. Then there is a pair (s,p) such that $(s,p) \in SGS(\mathcal{I})$, $(s,p) \notin LDA(\mathcal{I})$ and for all $(s',p') \succ_L (s,p)$, $(s',p') \in LDA(\mathcal{I})$ if and only if $(s',p') \in SGS(\mathcal{I})$. But then s prefers $SGS(\mathcal{I})$ to $LDA(\mathcal{I})$ because of Proposition 4.5. Therefore, (s,p) is a blocking pair in $LDA(\mathcal{I})$, which is not possible because of the stability of LDA (Proposition 5). □

Of course, student-optimality of SGS tells us that for any $\mathcal{I}$, $SGS(\mathcal{I}) \succeq_L LDA(\mathcal{I})$. Now, somewhat surprisingly, *SGS* and *LDA* are student-incomparable. Take $L = (s_1,p_1),(s_1,p_2),(s_2,p_2),(s_2,p_1)$ and assume that both s_1 and s_2 prefer p_2 to p_1. *Greedy* outputs $\{(s_1,p_1),(s_2,p_2)\}$. *LDA* outputs $\{(s_1,p_2),(s_2,p_1)\}$, which is better than *Greedy* for s_1 but worse for s_2. Still, we have the weaker property that *Greedy* cannot be better than *LDA* for all students: if s prefers $Greedy(\mathcal{I})$ to $LDA(\mathcal{I})$ then there must be a student s' who prefers $LDA(\mathcal{I})$ to $Greedy(\mathcal{I})$.

5 SUMMARY

We have defined three mechanisms that output a set of at most k student-project pairs from an input consisting of a master list of pairs and students' preferences over their acceptable projects. The list of properties of the obtained matchings and mechanisms are summarized in the table below.

	Greedy	LDA	SGS	all, L sc*
lexicographic optimality	+	-	-	+
stability	+	+	+	+
permutation strategyprf.	+	-	+	+
truncation strategyprf.	-	-	+	+
student optimality	-	-	+	+
dynamic monotonicity	-	+	+	+

Which mechanism to choose depends on the priority of the university. If what matters before all is the preferences of the committee expressed by the master list, then *Greedy* is probably the one to use: via the master list, it gives the committee the possibility of expressing the ability of a student to work better on a project than on another, and can also give priority to some projects or to some students over others. (This may explain why this mechanism is used in some universities.) But the price to pay is that students may dislike the outcome; especially, they may feel frustrated when they see that the outcome is not stable but that they are not allowed to exchange their projects. The SGS mechanism enjoys stability, strategyproofness, satisfies dynamic monotonicity, and is best for students, since its output is student-optimal among all stable matchings. The price to pay is a weaker importance given to the preference of the committee expressed by the master list; perhaps SGS is the mechanism to choose if the risk of students accepting offers from elsewhere is high. The LDA mechanism is a trade-off between both, as it seems to have a good balance between the master list and the students' preferences (although it can be sometimes worse than the output of *Greedy* for some students), and it outputs a stable matching, and satisfies dynamic monotonicity; but it is not strategyproof.

ACKNOWLEDGEMENTS

This work was funded in part by the Slovak Research and Development Agency under the contract APVV-17-0568, and by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute), and the ANR project AGAPE ANR-18-CE23-0013.

REFERENCES

[1] D.J. Abraham, R.W. Irving, and D.F. Manlove. The Student-Project Allocation Problem. In *Proceedings of ISAAC '03: the 14th Annual International Symposium on Algorithms and Computation*, volume 2906 of *Lecture Notes in Computer Science*, pages 474–484. Springer, 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: the 2nd International Conference on Computer and Electrical Engineering*, pages 375–379. IEEE, 2009.

[4] K. Cechlárová, L. Gourves, and J. Lesca. On the problem of assigning phd grants. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, pages 130–136, 2019.

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

[6] M. Goto, A. Iwasaki, Y. Kawasaki, R. Kurata, Y. Yasuda, and M. Yokoo. Strategyproof matching with regional minimum and maximum quotas. *Artif. Intell.*, 235:40–57, 2016.

[7] R.W. Irving, D.F. Manlove, and S. Scott. The stable marriage problem with master preference lists. *Discrete Applied Mathematics*, 156(15):2959–2977, 2008.

[8] K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation bounds for the student-project allocation problem with preferences over projects. *Journal of Discrete Algorithms*, 13:59–66, 2012.

[9] A. Kwanashie, R.W. Irving, D.F. Manlove, and C.T.S. Sng. Profile-based optimal matchings in the Student–Project Allocation problem. In *Proceedings of IWOCA 2014: the 25th International Workshop on Combinatorial Algorithms*, to appear, Lecture Notes in Computer Science. Springer, 2015.

[10] D.F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.

[11] D.F. Manlove and G. O'Malley. Student project allocation with preferences over projects. *Journal of Discrete Algorithms*, 6:553–560, 2008.