Assigning Course Schedules: About Preference Elicitation, Fairness, and Truthfulness

Martin Bichler, Soeren Merting
Department of Informatics, 85748 Garching, Technical University of Munich, Germany, {bichler, soeren.merting}@in.tum.de
Aykut Uzunoglu
Faculty of Business and Economics, 86135 Augsburg, University of Augsburg, Germany, aykut.uzunoglu@wiwi.uni-augsburg.de

Course assignment is a wide-spread problem in education and beyond. Often students have preferences for bundles of course seats or course schedules over the week, which need to be considered. The problem is a challenging distributed scheduling task requiring decision support. First-Come First-Served (FCFS) is simple and the most widely used assignment rule in practice, but it leads to inefficient outcomes and envy in the allocation. Recent theoretical results suggest alternatives with attractive economic and computational properties. Bundled Probabilistic Serial (BPS) is a randomized mechanism satisfying ordinal efficiency, envy-freeness, and weak strategy-proofness. This mechanism also runs in polynomial time, which is important for the large problem instances in the field. We report empirical results from a first implementation of BPS at the Technical University of Munich, which allows us to provide important empirical metrics such as the size of the resulting matching, the average rank, the profile, and the popularity of the assignments. These metrics were central for the adoption of BPS. In particular, we compare these metrics to Random Serial Dictatorship with bundle bids (BRSD). The BRSD mechanism is used to simulate the wide-spread First-Come First-Served (FCFS) mechanism and it allows us to compare FCFS (BRSD) and BPS. While theoretically appealing, preference elicitation is a major challenge when considering preferences over exponentially many packages. We introduce tools to elicit preferences which reduce the number of parameters a student needs to a manageable set. The approach together with BPS yields a computationally effective tool to solve course assignment problems with thousands of students, and possibly provides an approach for other distributed scheduling tasks in organizations.

Key words: Course Assignment, Perference Elicitation, Randomization, Field Study

Working Paper: Version December 10, 2018

1. Introduction

Course assignment is arguably one of the most wide-spread assignment problems where money cannot be used to allocate scarce resources. Such problems appear at most educational institutions. Matching with preferences has received significant attention in the recent years. While simple first-come first-served (FCFS) rules are still wide-spread, many organizations adopted matching mechanisms such as the deferred acceptance algorithm (Gale and Shapley 1962, Diebold et al.

2014) or course bidding (Sönmez and Ünver 2010, Krishna and Ünver 2008) to allocate scarce course seats. Although many course assignment problems are similar to the widely studied school choice problems with students private preferences for one out of many courses, other applications differ significantly. In particular, students are often interested in schedules of courses across the week. Assigning schedules of courses has been referred to as the *combinatorial assignment problem* (CAP) (Budish 2011). Similar problems arise when siblings should be assigned to the same schools in school choice (Abdulkadiroğlu et al. 2006), or couples in the context of the hospital residency matching (Ashlagi et al. 2014). Overall, the CAP can be seen as a general form of a distributed scheduling problem.

Although there is a huge body of literature on scheduling, the CAP is specific in a number of ways. First, we can only elicit ordinal preferences and no money must exchange hands. Second, students have private preferences over course schedules and we want to have mechanisms that incentivize students to reveal these preferences truthfully. Third, apart from efficiency, fairness of the allocation is an important concern in matching with preferences (Roth 1982). Fourth, the allocation of course schedules is a computationally hard (NP-hard) problem and for the problem sizes with hundreds of students an exact solution might not be tractable.

The need to assign course schedules rather than courses individually became apparent in an application of matching with preferences at the Technical University of Munich that we will discuss. The Department of Informatics is using the deferred acceptance algorithm for two-sided matching problems and random serial dictatorship for one-sided matching problems. These algorithms are used to assign seminars or practical courses, and every semester about 1500 students are being matched centrally (Diebold et al. 2014). For seminars and practical courses students need to get assigned one out of many courses offered per semester.

In the initial three semesters the situation is different. There are large courses with hundreds of students (e.g. on linear algebra or algorithms). These courses include a lecture and small tutor groups. Students need to attend one tutor group for three to four courses in each semester. These tutor groups should not overlap and they should be adjacent to each other such that students do not have a long commute for each of the tutor groups individually. For example, a student might want to have two tutorials in the morning and one after lunch on a particular day to reduce his commute time, and he would have a strong preference for this schedule over one where the tutorials are scattered across the week. In any case, students have timely preferences over course schedules that need to be considered, which makes it a combinatorial assignment problem. These problems are wide-spread in academia.

A first and seminal approach to address this challenging problem, the *approximate competitive equilibrium from equal incomes mechanism* (A-CEEI), was published by Budish (2011). In A-CEEI

students report their complete preferences over schedules of courses, the mechanism assigns a budget of fake money to each student that she can use to purchase packages (or schedules) of courses. Then an optimization-based mechanism computes approximate competitive equilibrium prices, and the student is allocated her most preferred bundle given the preferences, budgets, and prices. It is well known that serial dictatorships are the only strategy-proof and efficient mechanisms for multi-unit and also combinatorial assignment problems (Pápai 2001, Ehlers and Klaus 2003). A-CEEI is relaxing design goals such as strategy-proofness and envy-freeness to approximate notions, which makes it a remarkable and practical contribution to a fundamentally hard problem. The mechanism has been shown to be approximately strategy-proof, approximately envy-free, and Pareto efficient. Budish et al. (2017) reports the empirical results at the Wharton School of Business. In addition, Budish and Kessler (2017) summarize the results of lab experiments.

The work was breaking new ground, but the A-CEEI mechanism is also challenging. First, it is not guaranteed that a price vector and course allocation exists that satisfies all capacity constraints. This is not surprising given that prices are linear and anonymous. Second, the problem of computing the allocation problem in A-CEEI is PPAD-complete and the algorithms proposed might not scale to larger problem sizes required in the field (Othman et al. 2016). Third, students might not be able to rank-order an exponential set of bundles, which is a well-known problem (aka. missing bids problem) in the literature on combinatorial auctions (with money) (Milgrom 2010, Bichler et al. 2011, 2014). The latter is a general problem in CAP not restricted to A-CEEI, which we will discuss in much more detail below.

Randomization can be a powerful tool in the design of algorithms, but also in the design of economic mechanisms. Nguyen et al. (2016) recently provided two randomized mechanisms for one-sided matching problems, one with cardinal and one with ordinal preferences for bundles of objects. The mechanism for ordinal preferences is a generalization of probabilistic serial (Bogomolnaia and Moulin 2001a), called Bundled Probabilistic Serial (BPS). Nguyen et al. (2016) show that this randomized mechanism is ordinally efficient, envy-free, and weakly strategy-proof. These appealing properties come at the expense of feasibility, but the constraint violations are limited by the size of the packages. In course assignment problems the size of the packages is typically small (e.g., packages with three to four tutor groups) compared to the capacity of the courses or tutor groups (around 30 seats or more). There is no need for prices or budgets, and computationally the mechanism runs in polynomial time, which is important for large instances of the course allocation problem that can frequently be found. This makes BPS a practical approach to many problems that appear in practice.

1.1. Contributions

We report on a first large-scale field study of BPS and address important problems in the implementation of mechanisms for the combinatorial assignment problem that are beyond a purely theoretical treatment. In particular, preference elicitation is a central concern in combinatorial mechanisms with a fully expressive bid language and we provide a practical approach that addresses the combinatorial explosion of possible packages for many applications. Theoretical contributions of assignment mechanisms largely focus on envy-freeness and efficiency as primary design desiderata. We report properties of matchings such as their size, their average rank, the probability of matching, the profile, and the popularity. These properties are of central importance for the choice of mechanisms. For IS designers it is important to understand the trade-offs with other mechanism, in particular with the wide-spread FCFS.

Implementing and testing new IS artifacts for coordination in organizations is challenging and we are grateful for the possibility to run a large-scale field experiment at the Department of Informatics of the Technical University of Munich (TUM). This is particularly true for a non-trivial mechanism such as BPS, which involves advanced optimization and randomization. Yet, we can report on the assignment of 1439 students in the summer term 2017 to 67 tutor groups for 4 classes and the assignment of 1778 students in the winter term 2017/2018 to 66 tutor groups for 4 classes using BPS.[1] Based on this data the department has adopted the new mechanism for good.

For such a large application we could not elicit preferences of students for BPS and let them participate in FCFS simultaneously. Instead we simulated FCFS via a version of Random Serial Dictatorship that allows for bundles (BRSD), which is of independent interest as an assignment mechanism. In our numerical experiments we simulated FCFS via a large number of random order arrivals in BRSD using the preferences elicited in BPS and average across all of them. This approach allows for a comparison between BPS and BRSD (FCFS) on equal footing.

FCFS only collects limited information about the preferences of participants, a single package only. Mechanisms for the combinatorial assignment problem allow participants to specify preferences for all possible packages. However, a fully enumerative bid language requires participants to submit preferences for an exponential set of packages which is impractical. Preference elicitation and user interface design have long been a topic in IS research (Santos and Bariff 1988, Lee and Benbasat 2011). We contribute an approach that is applicable in a wide array of CAP applications where timely preferences matter. We elicit a small number of parameters about breaks and preferred times and days of the week. Together with some prior knowledge about student preferences this allows us to score and rank-order all possible packages. Students could iteratively adapt the

[1] Not all students submitted a non empty preference list. Therefore, we consider in our evaluation only 1415 students in summer term and 1736 students in winter term.

parameters and the ranking, which then served as an input for BPS. While such ranking algorithms will differ among types of applications, adequate decision support that aids the ranking of exponentially many packages is a crucial prerequisite to actually achieve the benefits of combinatorial assignment in real-world applications.

In our empirical analysis, we show that BPS has many advantages over BRSD in all of the properties introduced earlier. While the differences in these criteria are small, envy-freeness turns out to be the most compelling advantage of BPS. The level of envy that we find in BRSD is substantial in spite of the limited complementarities in student preferences, who are only interested in packages with at most four tutor groups. This has to be traded off with the simplicity of FCFS. Overall, we *empirically test and illustrate theory* that has been developed only recently. We show that randomized matching mechanisms together with appropriate decision support tools are a powerful new IS design recipe for daunting coordination problems in organizations. Thus, we contribute to the traditional IS research stream in decision support and design science research, but introduce new methods and applications (Banker and Kauffman 2004).

2. Combinatorial Assignment Problems

Let us now define the combinatorial assignment problem (CAP) in the context of course assignment applications, desirable properties, and randomized mechanisms.

2.1. Assignment Problems

Assigning objects to agents with preferences but without money is a fundamental problem referred to as *assignment problem with preferences* or *one-sided matching with preferences*. We will use the term assignment or matching interchangeably. In course assignment, students express ordinal preferences which need to be considered in the assignment. A *one-sided one-to-many course assignment problem* consists of a finite set of n students (or agents) S and a finite set of m courses (or objects) C with the *maximum capacities* $q = (q_1, q_2, \ldots, q_m)$.

In the *combinatorial assignment problem* in the context of course allocation, every student $i \in S$ has a complete and transitive preference relation $\succeq_i \in \mathcal{P}$ over subsets (or bundles) of elements of C. A preference profile $\succeq = (\succeq_1, \cdots, \succeq_n) \in \mathcal{P}^{|S|}$ is an n-tuple of preference relations. For most of the paper we will assume strict preferences, but we discuss indifferences in the conclusions. We can model the demand of the students with binary vectors $b \in \{0,1\}^m$, where $b_j = 1$ if course j is included in b. We define the size of a bundle b with $size(b) = \sum_{j=1}^{m} b_j$, the number of different courses included in the bundle. Let B be the set of all feasible bundles b. Let x_{ib} be a binary variable describing if bundle b is assigned to student i. Then we can model the demand and supply as linear constraints. The supply constraints make sure that the capacity of the courses are not

exceeded, and the demand constraints determine that each student can win at most one bundle.

$$\sum_{i \in I, b \in B} x_{ib} b_j \leq q_j \qquad j \in C \qquad \text{(supply)}$$

$$\sum_{b \in B} x_{ib} \leq 1 \qquad \forall i \in S \qquad \text{(demand)}$$

$$x_{ib} \in \{0,1\} \qquad \forall i \in S, b \in B \qquad \text{(binary)}$$

Courses in our application are actually tutor groups and each tutor group belongs to one of ℓ classes. Students in our application can only select bundles with at most one tutor group in each of these classes. For example, a student might select a bundle with a course seat in a tutor group for mathematics on Monday at 1 pm, and another tutor group in software engineering two hours later, but no additional tutor group in mathematics or software engineering in this bundle. As a result, the possible size of a bundle b is $size(b) \leq \ell \ll m$. The Web interface takes care that students only submit valid bundles, which have at most one tutor group for each of the ℓ classes and a size less than or equal to ℓ.

A *deterministic combinatorial assignment* (deterministic matching) is a mapping $M \subset S \times B$ of students S and bundles B of courses C. $\mathcal{M}$ describes the set of all deterministic matchings. A matching is *feasible* if it is a feasible integer solution to the constraints demand and supply. *Random combinatorial assignments* (random matchings) are related to fractional assignments with $0 \leq x_{ib} \leq 1$ and random assignment mechanisms can be used to fractionally allocate bundles of course seats to students.

For (non-combinatorial) assignment problems with single-unit demands the Birkhoff-von-Neumann theorem (Birkhoff 1946, Von Neumann 1953) says that every fractional allocation can be written as a unique probability distribution over feasible deterministic assignments. That is, any random assignment can be implemented as a lottery over feasible deterministic assignments, such that the expected outcome of this lottery equals the random assignment. One can describe a random assignment as a bistochastic matrix, where p_{ic} is the probability that student i is assigned to course c. The Birkhoff-von-Neumann theorem shows that such a bistochastic matrix can be decomposed into a convex combination of permutation matrices, which describe feasible deterministic assignments. However, the Birkhoff-von-Neumann theorem fails when bundles of course seats need to be assigned. Nguyen et al. (2016) generalize this result and show that any fractional solution respecting the demand and supply constraints can be implemented as a lottery over integral allocations that violate the supply constraints only by at most $\ell - 1$ course seats.

2.2. Design Desiderata

Efficiency, envy-freeness, and strategy-proofness are design desiderata of first-order importance typically considered in the theoretical literature on deterministic assignment problems. For randomized mechanisms one has to reconsider these design desiderata and we will briefly introduce relevant definitions in this section. Stochastic dominance (SD) is the key concept among all of these definitions as it provides a natural way to compare random assignments. Let Δ describe the set of all possible random matchings. With p_i we refer to the assignment of student i in the random matching p, and denote with p_{ib} the probability that student i gets allocated bundle b. We will omit the subscript i when it is clear which student is meant. Given two random assignments $p, q \in \Delta$, student i *SD-prefers* p to q if, for every bundle b, the probability that p yields a bundle at least as good as b is at least as large as the probability that q yields a bundle at least as good as b.

Definition 1 (SD-prefer). A student $i \in S$ *SD-prefers* an assignment $p \in \Delta$ over $q \in \Delta$, $p \succeq_i^{SD} q$, if

$$\sum_{b' \succeq_i b} p_{ib'} \geq \sum_{b' \succeq_i b} q_{ib'}, \forall b \in B \tag{2}$$

In other words, a student i prefers the random assignment p to the random assignment q if p_i stochastically dominates q_i. Note, that $\succeq^{SD}$ is not a complete relation. That is there might be assignments p and q, which are not comparable with this relation. First-order stochastic dominance holds for all increasing utility functions and implies second-order stochastic dominance, which is defined on increasing concave (risk-averse) utility functions. In other words, risk-averse expected-utility maximizers prefer a second-order stochastically dominant gamble to a dominated one.

Nguyen et al. (2016) show that a lottery over allocations of bundles induces probability shares over these bundles that satisfy demand and supply constraints. Thus a lottery coincides with a fractional solution to both constraints. However, a fractional solution respecting demand and supply does not need to have a lottery over deterministic assignments.

One desirable property of matchings is *(Pareto) efficiency* such that no student can be made better off without making any other student worse off. A *deterministic* matching M is *efficient* with respect to the students if there is no other feasible matching M' such that $M'(i) \succeq_i M(i)$ for all students $i \in S$ and $M'(i) \succ_i M(i)$ for some $i \in S$. One can generalize this to random matchings and lotteries:

Definition 2 (Efficiency). A random assignment $p \in \Delta$ is *ex post efficient*, if p can be implemented into a lottery over Pareto efficient deterministic assignments. A random assignment $p \in \Delta$ is *ordinally efficient*, if there exists no random assignment q such that q stochastically dominates p, i.e. $\nexists q \in \Delta : \forall i \in S : q \succeq_i^{SD} p$ and $\exists i \in S : q \succ_i^{SD} p$.

Ordinal efficiency comes from the Pareto ordering induced by the stochastic dominance relations of individual students. It can be shown that ordinal efficiency implies ex post efficiency (Bogomolnaia and Moulin 2001a).

Fairness is another important design goal. A basic notion of fairness for randomized assignments is the *equal treatment of equals*, i.e. students with identical preferences receive identical (symmetric) random allocations. Envy-freeness is stronger.

Definition 3 (Envy-Freeness). A random assignment $p \in \Delta$ is *(strongly) SD-envy-free*, if $\forall i, j \in S : p_i \succeq_i^{SD} p_j$. We call p *weakly SD-envy-free*, if $\nexists i, j \in S : p_j \succ_i^{SD} p_i$.

SD-envy-freeness means that student i weakly SD-prefers the random matching she is faced with to the random assignment offered to any other student, i.e., a student's allocation stochastically dominates the outcome of every other student. For weak SD-envy freeness it is only demanded that no student's allocation is stochastically dominated by the allocation of another student. SD-envy-freeness implies equal treatment of equals.

A *randomized assignment mechanism* is a function $\psi : \mathcal{P}^{|S|} \to \Delta$ that returns a random matching $p \in \Delta$. The mechanism $\psi(\succeq) = p$ is *ordinally efficient* if it produces ordinally efficient allocations. In terms of fairness, one could aim for a matching where equals are treated equally. We call a randomized matching mechanism ψ *symmetric*, if for every pair of students i and j with $\succeq_i = \succeq_j$ also $p_i = p_j$. This means that students who have the same preference profile also have the same outcome in expectation. A randomized mechanism is *envy-free* if it always selects an envy-free matching.

An important property of a mechanism is *strategy-proofness*. This means, that there is no incentive for any student not to submit her truthful preferences, no matter which preferences the other students report. A deterministic assignment mechanism χ is *strategy-proof* if for any $\succeq \in \mathcal{P}^{|S|}$ with $i \in S$ and $\succeq_i' \in \mathcal{P}$ we have $\chi_i(\succeq) \succeq_i \chi_i(\succeq_i', \succ_{S \setminus \{i\}})$. It has been shown that participants in strategy-proof mechanisms such as the Vickrey auction do not necessarily bid truthfully in practice. Therefore, there was a recent discussion about obvious strategy-proofness of extensive form games (Li 2017). Intuitively, a mechanism is obviously strategy-proof iff the optimality of truth-telling can be deduced without contingent reasoning. Pycia and Troyan (2016) show that RSD is a unique mechanism that is obviously strategy-proof, efficient, and symmetric in mechanisms without transfers.

For randomized mechanisms we need to adapt the definitions. A random assignment mechanism is (strongly) SD-strategy-proof if for every preference profile $\succeq$, and for all $i \in S$ and $\succeq_i'$ we have $\psi(\succeq_i, \succeq_{-i}) \succeq_i^{SD} \psi(\succeq_{i'}, \succeq_{-i})$. A random assignment rule ψ is *weakly* SD-strategy-proof if for every preference profile $\succeq$, there exists no $\succeq'$ *for some* student $i \in S$ such that $\psi(\succeq_{i'}, \succeq_{-i}) \succeq_i^{SD} \psi(\succeq_i, \succeq_{-i})$. That is, there may not be any student i, who strictly prefers $\psi(\succeq_{i'}, \succeq_{-i})$ over the truthful outcome,

but there may be students i who neither prefer $\psi(\succeq_{i'}, \succeq_{-i})$ nor $\psi(\succeq_i, \succeq_{-i})$. This can happen as the $\succeq^{SD}$-relation is not complete. We will omit the prefix SD for brevity in the following. Note that there are also weaker notions of strategy-proofness for randomized mechanisms developed in the field of probabilistic social choice that we do not consider in this article. These notions are based on different ways of how to compare lotteries. Interested readers are referred to Brandt (2017).

In section 4.1 we introduce a number of additional design goals that often matter in the practice and that we analyze empirically.

2.3. Assignment Mechanisms

A lot is known about assignment problems with single-unit demand. Random Serial Dictatorship (RSD) selects a permutation of the agents uniformly at random and then sequentially allows agents to pick their favorite course among the remaining ones. Gibbard (1977) showed that random dictatorship is the only anonymous and symmetric (in the sense of equal treatment of equals), strongly SD-strategy-proof, and ex post efficient assignment rule when preferences are strict. Pycia and Troyan (2016) prove that RSD is a unique mechanism that is obviously strategy-proof, efficient, and symmetric in mechanisms without transfers. In line with this recent result, Ashlagi and Gonczarowski (2015) show that stable matching mechanisms are not obviously strategy-proof.

However, RSD is not always ordinally efficient, only ex post efficient (Bogomolnaia and Moulin 2001b). Zhou (1990) actually showed that no random mechanism for assigning objects to agents can satisfy strong notions of strategy-proofness, ordinal efficiency, and symmetry simultaneously with more than three objects and agents. So, we also cannot hope for these properties in combinatorial assignment problems. RSD can also be applied to the combinatorial assignment problem. The Bundled Random Serial Dictatorship (BRSD) orders the students randomly and assigns the most preferred bundle which is still available to each student in this order. Although the package preferences take some toll on the runtime it is still very fast.

First-come first-served (FCFS) can be seen as a serial dictatorship. Students login at a certain registration and then reserve the most preferred bundle of courses that is still available. Although the arrival process is not uniform at random, students have little control over who arrives first. While there is a certain time when the registration starts, hundreds of students log in simultaneously to get course seats and it is almost random who arrives first. We will simulate FCFS via BRSD and run the algorithm repeatedly to get estimates for performance metrics of FCFS.

Probabilistic Serial (PS) (Bogomolnaia and Moulin 2001b) produces an envy-free assignment with respect to the reported unit-demand preferences, and it is ordinally efficient, but it is only weakly SD-strategy-proof. Bundled Probabilistic Serial (BPS) by Nguyen et al. (2016) is a generalization of PS to the combinatorial assignment problem. BPS computes a fractional solution via a

generalization of the PS mechanism. The BPS mechanism is also ordinally efficient, envy-free, and weakly strategy-proof if preferences are strict, which we will discuss as an issue in the conclusions.

Informally, in BPS all agents *eat* their most preferred bundle in the time interval $[0,1]$ simultaneously with the same speed as long as all included objects are available. As soon as one object is exhausted, every bundle containing this object is deleted and the agents continue eating the next available bundle in their preference list. The duration with which every bundle was eaten by an agent specifies the probability for assigning this bundle to this agent.

Algorithm 1: Pseudocode of BPS.

Input: Preferences $(\succeq_i)_{i \in S}$

$t = 0$

$x_{ib} = 0,\ \forall i \in S, b \in B$

while $t < 1$ **do**

  $D = \emptyset$

  $dem_j = 0,\ \forall j \in C$

  forall $i \in S$ **do** choose first valid bundle $b \in \succeq_i$: $D \leftarrow b$

  forall $b \in D$ **do**

    forall $j \in b$ **do** $dem_j{+}{+}$

  $\Delta = min\left\{ \frac{dem_j}{q_j} \mid j \in C \right\}$

  $t += \Delta$

  $\Delta^* = \Delta - (t - min\{1, t\})$

  forall $i \in S$ **do** $x_{ib} += \Delta^*$

  forall $j \in C$ **do**

    $q_j -= \Delta^* \cdot dem_j$

    if $q_j = 0$ **then** $\forall b \in B : j \in b$: delete b

Output: Allocation $x^* = (x_{ib})_{i \in S, b \in B}$

2.4. Implementing Random Assignments

Unfortunately, in contrast to the result of PS, the outcome of BPS is not implementable into a lottery of deterministic matchings in general if $\ell > 1$. To circumvent this, one can either scale x^* by a factor $\alpha \in [0,1]$ such that the decomposition becomes possible (Lavi and Swamy 2011) or one allows for the relaxation of some constraints. Nguyen et al. (2016) present a mechanism to decompose the BPS solution into a lottery over deterministic matchings, which over-allocate each course by at most $\ell - 1$ seats, i.e. the demand constraints are fulfilled and only the supply constraints are relaxed.

In the polynomial time *lottery algorithm* (see Algorithm 3), we find at most $d+1$ integral points, the convex hull of which is arbitrarily close to the fractional solution x^*, which we get from BPS. The lottery algorithm then returns a lottery over these $d+1$ integral vectors, which is close to x^* in expectation. Variable d describes the dimensions of the problem. In this lottery algorithm, we use a subroutine to return an integer point $\bar{x}$ such that $u^\tau \bar{x} \geq u^\tau x^*$. This subroutine is called *iterative rounding algorithm* (IRA) and proceeds as described in Algorithm 2.

Algorithm 2: Pseudocode of the iterative rounding algorithm.

1a: Delete all $x_i = 0$, $x_i = 1$, update the constraints and go to 1b.

1b:

If there is no $x_i \in \{1, 0\}$ one can find at least one supply-constraint with

$$\sum_{i \in \mathcal{S}} \sum_{b: j \in b} b_j \lceil x_{ib} \rceil \leq q_j + \ell - 1$$

Delete those constraints and go to 2.

2

Solve $max\{u^\tau x \mid (demand), (supply), x \in \mathbb{R}_{\geq 0}\}$

if *all* $x_i \in \{0, 1\}$ **then** return x

else go to 1a

Now, we can discuss the lottery algorithm (see Algorithm 3). Let $\mathcal{B}(x^*, \delta) = \{x \mid |x^* - x| \leq \delta\} \subseteq \{x \in \mathbb{R}_{\geq 0} \mid (Demand)\}$ with $\delta > 0$. The parameter δ in $\mathcal{B}(x^*, \delta)$ determines some space around x^* such that the demand constraint $\sum_{b \in B} x_{ib} \leq 1$ is not violated. It is always possible to determine such a δ. If there is no slack in the demand constraints one has to scale down the fractional solution x^*. Afterwards one has to adjust the allowed error ε such that after scaling and decomposition the original ε is still fulfilled. Here, $|x - y|$ describes the Euclidean distance between two vectors x and y.

Figure 1 shows a graphical representation of one algorithm iteration. In each iteration the algorithm decreases the distance between y and x^* by adding a new integral solution to the solution set Z and terminates when the distance between y and x^* is smaller than ε. That is, we consider y as a good approximation for x^* and return the support of y. The algorithm tries to get x^* covered by the convex hull of Z ($conv(Z)$). All solutions in Z that are not part of the support of y, calculated in the quadratic optimization problem (QOP) in step 2, are deleted (step 3). Thus, although we add a new integral solution to Z in each iteration, the size of Z never grows above $d+1$, since as long as $y \neq x^*$, y always has to be on a face of $conv(Z)$. Hence, the support of y consists of at most d solutions. Step 4 ensures that we search in the right direction for new integral solutions. As a

Algorithm 3: Pseudocode of the lottery algorithm.

Input: Fractional solution x^*

1: Set $Z = \{\text{IRA}(x^*)\}$, i.e., find integer solution via IRA.

2:
  $y = argmin\{|x^* - y| \mid y \in conv(Z)\}$
  if $|x^* - y| < \varepsilon$ **then** END

3: Choose $Z' \subset Z$ of size $|Z'| \leq d$ and $y \in conv(Z') : \; z = x^* + \delta \frac{x^* - y}{|x^* - y|}$

4: Find optimal integral z' s.t. (Demand),(Supply) and $(x^* - y)^\intercal z' \geq (x^* - y)^\intercal z$ via IRA

5: $Z = Z' \cup \{z'\}$ and go back to 2.

Output: Convex combination of final y

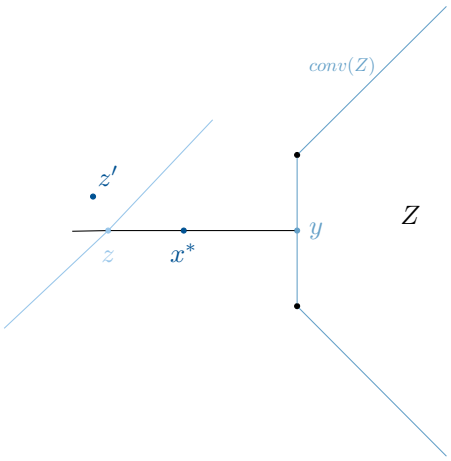

Figure 1 **Graphical representation of one iteration of the lottery algorithm.**

side product the QOP also calculates the coefficients $\lambda^{(k)}$ for the convex combination and we have $x^* \approx y = \sum_{k=1}^{|Z|} \lambda^{(k)} x^{(k)}$, for $x^{(k)} \in Z$.

3. Preference Elicitation

This section focuses on the preference elicitation, which is important given the exponential set of possible bundles students might be interested in. We first introduce the environment and the problem for students, before we discuss different approaches to elicit their preferences.

3.1. Background on the Application

The Department of Informatics has been using stable matching mechanisms for the assignment of students to courses since 2014 (Diebold and Bichler 2017, Diebold et al. 2014). The system provides a web-based user interface and every semester almost 1500 students are being matched to lab courses or seminars via the deferred acceptance algorithm for two-sided matching or random serial dictatorship for one-sided matching problems.

In the context of the study reported in this paper, the web-based software was extended with BPS, the lottery mechanism for decomposing fractional solutions, and BRSD. 1439 Students in computer science and information systems in their second semester participated in the matching during the summer term 2017 and they could choose tutorial groups from several courses including linear algebra, algorithms, software engineering, and operations research. A computer science student could have preferences for up to 5760 ($= 10 \cdot 24 \cdot 24$) bundles[2] and an information systems student could have preferences for up to 5184 ($= 9 \cdot 24 \cdot 24$) bundles.[3] During the winter term 2017/2018, 1778 computer science and information systems students in their third semester participated in the matching and could choose bundles of tutor groups out of four classes. A computer science student could have more than 700,000 different bundles.[4]

3.2. Automated Ranking of Packages

A naive approach would be to let the students rank bundles on their own by choosing the time slots they want to have in their preference list. This would take a lot of time and lead to a substantial missing bids problem. We developed an algorithm that allows to rank-order all possible packages based on a few parameters that students need to specify. For this, we can leverage prior knowledge about timely preferences of students for schedules of tutorials and lectures.

Students' preferences mainly concern their commute and the possibility to free large contiguous blocks of time (e.g., a day or a half-day) that they can plan for other activites (e.g., a part-time job). In larger cities such as Munich, the time that students spend for commuting is significant. Also long waiting times between courses are perceived as a waste of time as it is often hard for them to work productively in several one- or two-hour breaks without appropriate office facilities available. For example, if a student had a tutorial on linear algebra in the morning, a lunch break, and then the tutorials for algorithms and software engineering in the afternoon of the same day with the minimal time for breaks specified, this would be considered ideal. The desired length for breaks between tutorials and for the lunch break are considered parameters with default values in the preference elicitation.

Figure 2 shows the initial page where a student can select the courses of interest. On this page students choose the lectures and tutorials they are interested in. The selected lectures will be considered in the bundle generation as constraints, i.e. if a time slot of a tutorial overlaps with the time of a selected lecture, then it will no longer be considered in order to allow students to participate in the lecture. In a second step, the student marks available time ranges in a *weekly*

[2] Consisting of the courses: linear algebra, algorithms, software engineering.

[3] Consisting of the courses: operations research, algorithms, software engineering.

[4] The computer science students need tutorials from all four classes ($< 22 \cdot 25 \cdot 26 \cdot 52$).

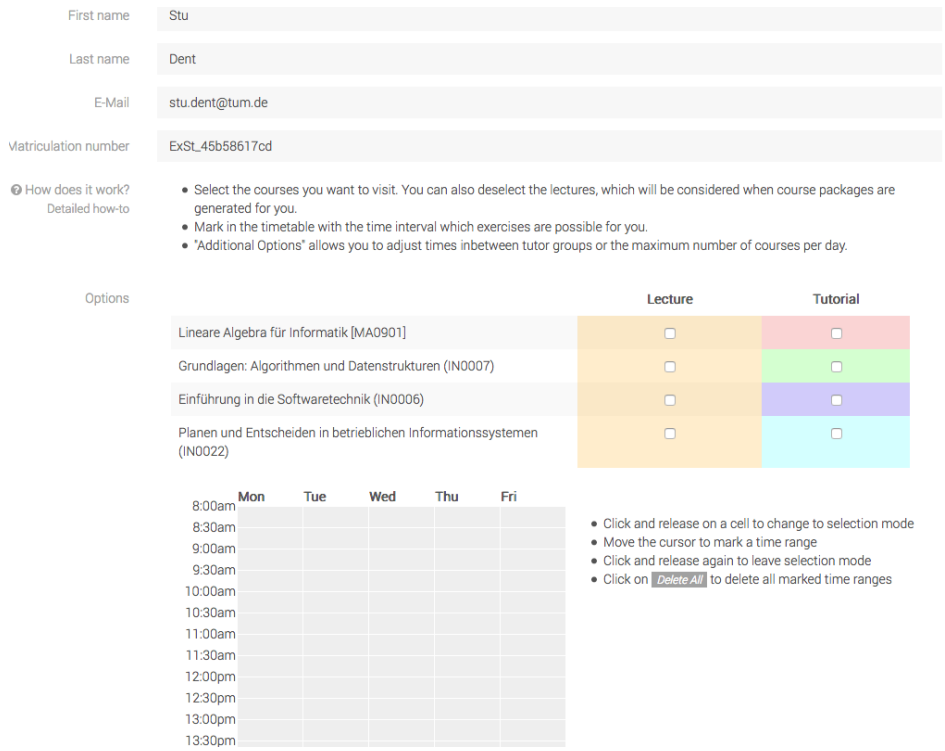

Figure 2 User Interface to Select Courses

schedule (see Figure 3). The day is partitioned into weekdays and time blocks of 30 minutes from 8:00 AM to 8:30 PM. If a tutorial is selected, all time slots of this tutorial will be highlighted with a specific color. Thus, students learn when the tutorials and lectures of interest take place.

A student can set a minimal amount of time for a lunch break and a minimal amount of time in-between two events (default value is 15 minutes). We also allow students to provide weights $\{1,\ldots,5\}$ for the different days. That is, the students can express preferences over the days.

The preferences elicited on this screen are input for an algorithm that uses prior knowledge about student preferences to rank-order all possible packages. The algorithm first generates bundles that satisfy all constraints and then ranks them. Finding the bundles that do not violate constraints (e.g., lectures to be attended) of the students can be cast as a *constraint satisfaction problem*. After the feasible bundles are generated, we rank these bundles. For this we assign a score to each bundle that considers

- how many days a student needs to come to the university per week in total,
- the preference ordering over the days,
- the total time a student has to stay at the university each day, and
- the length of the lunch breaks between courses.

The score for a package b of courses across the week is the sum of the daily score ($score(b,d)$) for all weekdays d. The daily score is computed as

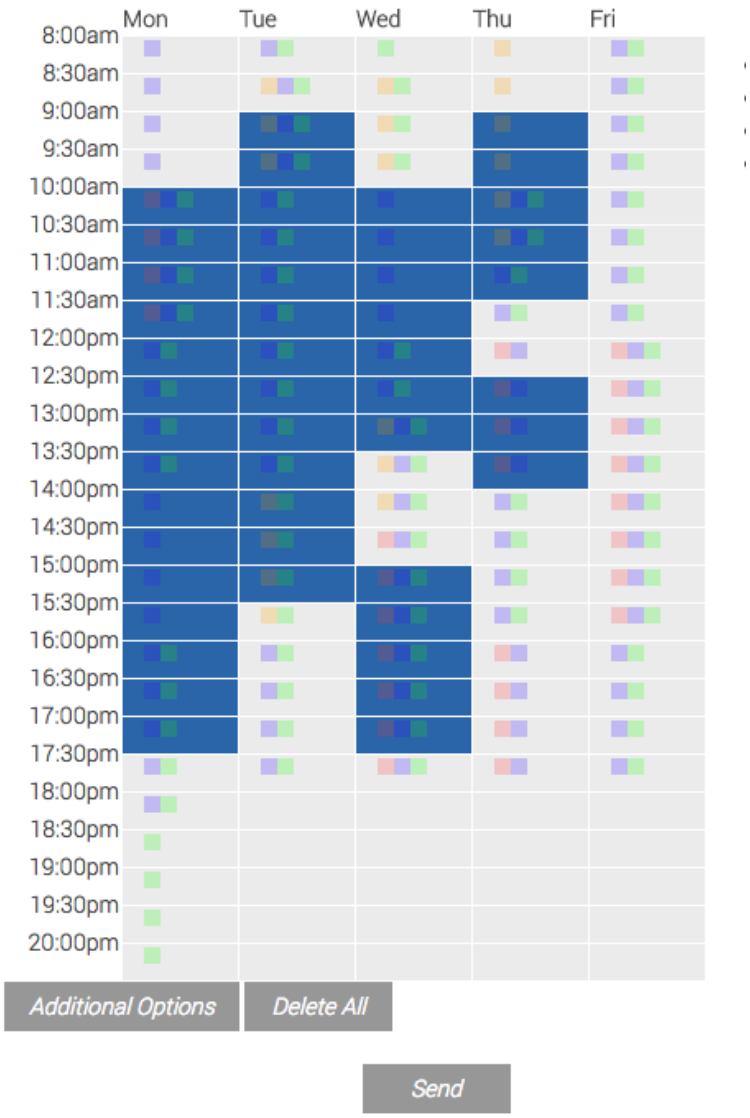

Figure 3 The Week Schedule

$$score(b, day) = \left(\frac{w(b, day)}{sp(b, day)} \cdot f(sp(b, day)) + br(b, day) \right) \cdot prio(day) \tag{3}$$

This score is scaled between 0 and 27.5 at a maximum and it considers how well the day is utilized with courses. Ideally, a student would have all his tutorials on a single day, his most preferred day, have a 1-hour lunch break and a minimal time for breaks inbetween courses. This would yield 27.5 points.

The time spent at the university per day $sp(b, day)$ is considered relative to the time a student attends courses on that day ($w(b, day)$). These courses include tutorials and lectures. The ratio is used to weigh the score for a day ($f(sp(b, day))$). This way a day where students do not spend more time in breaks than the minimum number of minutes specified maximizes the score. The function

$f(\cdot)$ assigns 1 point for up to 2 hours spent at the university on a day ($sp(b, day) \leq 2$), 2 points for up to 4 hours, 6 points for up to 3 hours, 4 points for up to 8 hours, but only 2 points for days where a student is between 8 and 10 hours at the university. Longer schedules are not permitted.

A second component in the daily score ($score(b, d)$) is the lunch break. A 1-hour break was considered best. The scoring function $br(\cdot)$ would assign 0 points for lunch breaks less than 30 minutes, 1 point for 30-45 minutes, 1.5 points for 45-60 minutes, 2 points for 60-75 minutes, and again a low number of points for longer breaks. Students could also exclude schedules with a break less than a certain time, say 30 minutes.

If the student does not have to visit the university at day d, he gets a fixed score of 30 for day d. The daily scores are then multiplied by the priority of the day [1..5]. The overall score of a bundle b is the sum of the $score(b, d)$ for all weekdays. As a result of this scoring rule, the more days the student can stay at home, the higher is the score of this bundle. As a simplified example, if a student had to come to the university on three different days to attend one course only, this bundle would get a score of less than 25, while if he could attend all courses on a single day with minimal breaks, this will get an overall score of more than 80 (for these three days).

In other words, the scoring rule will place bundles, that use a minimal number of days (ideally the most preferred days) with only a few breaks but a one hour lunch break on top of the preference list. This would minimize the commute time and maximize the contiguous time a student can devote to learning or work. If the breaks between courses grow larger or courses take place on different or more days, this decreases the score. Ties are not impossible but almost never occur such that the algorithm typically generates a strict ranking of the feasible packages.

Even if it is a fair assumption that students have quite homogeneous preference structures, there might be some special cases we cannot cover with such a scoring rule. Therefore we give the students the possibility to adjust the outcome of this scoring procedure. On the ranking page, we display the 30 top rated pre-ranked bundles and the students can adapt this ranking manually, go back to the previous screen and adapt the input parameters, or just accept the ranking with a single click (see Figure 4). Note that $\approx 90\%$ of the students received one of their top ten ranked packages and only a few students received a package with a rank less than 30. So, if a student inspects and confirms the ranking of the first 10-30 packages, this covers the most important quantile of the overall ranking list. We generated a ranking over 200 bundles for each student in advance based on the pre-specified parameters and further preferences only if necessary.

So far, we described the user interface for the winter term 2017/18. The user interface in the summer term 2017 required students to explicitly drag and drop the pre-ranked packages on a screen. This was considered tedious such that in the winter term the generated ranking was suggested to students right away without any drag-and-drop activies required and could be confirmed

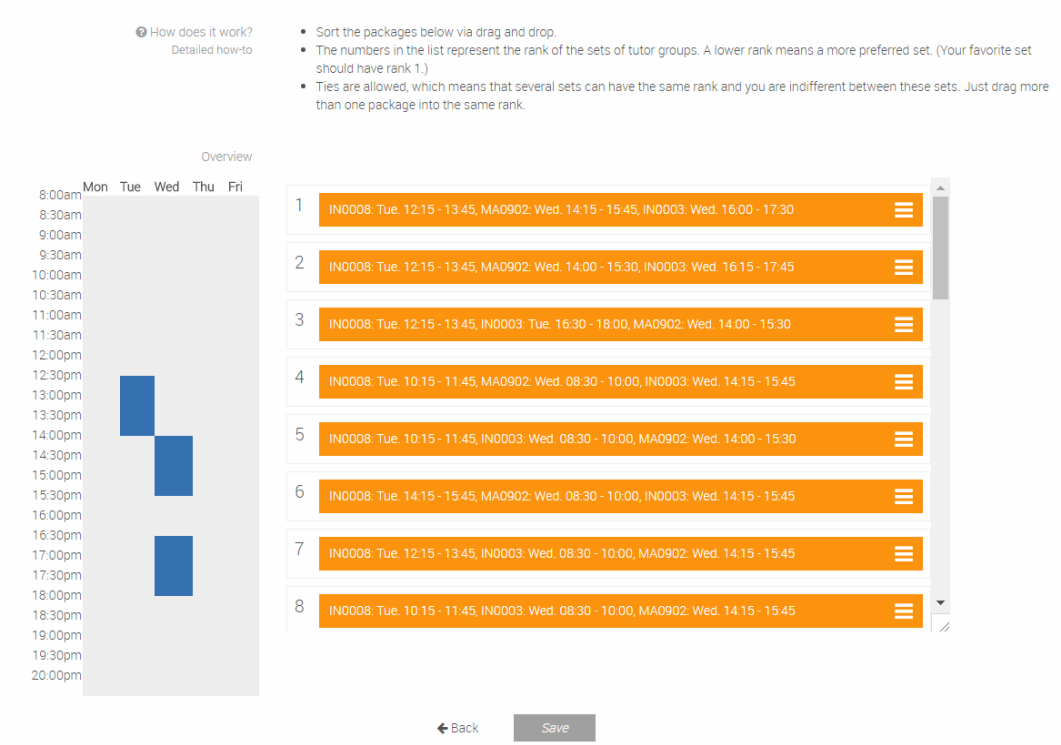

Figure 4 **Page with top-ranked packages**

without much effort. The main web page and the main steps a student had to take are summarized in Figure 5.

3.3. Challenges of Course-Level Scoring

Ranking an exponential set of packages is a general issue in course assignment problems, and one might ask if alternative methods are available. Budish et al. (2017) describes the preference elicitation used at the Wharton School of Business. Students could report cardinal item values on a scale of 1 to 100 for any course they were interested in taking. In addition, they could report adjustments for pairs of courses, which assigned an additional value to schedules that had both course sections together. Courses were then scored and transformed into an ordinal ranking over feasible schedules. The authors argue that they felt that "adding more ways to express non-additive preferences would make the language too complicated." Wharton also provided a decision support tool listing the 10 most-preferred bundles, which allowed students to inspect top-ranked schedules and modify the cardinal values.

Two problems make this method challenging to apply. First, the ranking is sensitive to minor changes in the weights, which is a well-known issue in multi-criteria decision making with additive value functions. Evaluation is characterized by a substantial degree of random error, and the amount of error tends to increase as the decision maker attempts to consider an increasing number

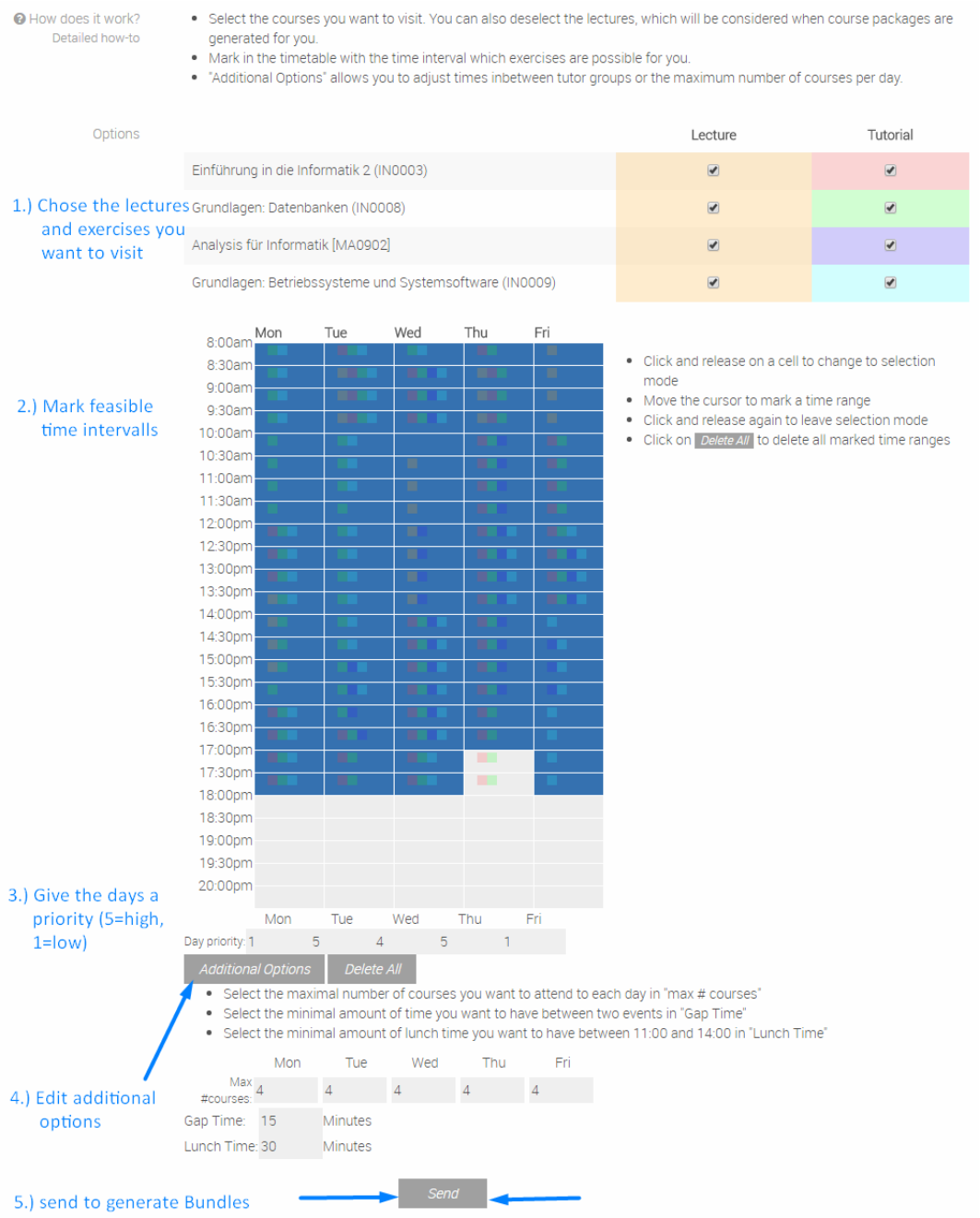

Figure 5 **Process to rank-order packages**

of attributes (or courses in our case) (Bowman 1963, Fischer 1972). Difficulties in the calibration of scores for each course can lead to substantial differences in the resulting ranking.

Second, and more importantly, significant non-linearities arise due to the timely preferences of students in the assignment of tutorials, making it impossible to describe the preferences via a course-level utility function as proposed by Budish et al. (2017). Even if three tutorials get a high number of points, this does not mean that their combination is preferable by a student as these tutorials

might be on different days or have long breaks inbetween. To see this, we translated the ranking of packages into a set of inequalities with weights (w) as variables. Following revealed preference theory (Mas-Colell et al. 1995), we use these inequalities to understand whether there is any set of weights that would allow to describe the ranking using a utility function $\sum_{i \in C} b_i w_i + \sum_{\substack{i,j \in C \\ j>i}} b_i b_j w_{ij}$. The function $r(b)$ describes the rank of a bundle, while b is a binary parameter vector with each component $b_i \in \{0,1\}$ showing whether a course $i \in C$ is part of a package or not. The objective function minimizes the sum of error variables ε in REV. If there is any set of weights that could reflect the ranking of packages in our experiments without these error variables, the resulting optimal objective value would be zero. For every violation of a constraint one has to increase the respective error variable to a positive value.

$$
\begin{aligned}
\underset{\text{s.t.}}{\text{Min}} \quad & err(\varepsilon) = \sum_{b \in B} \varepsilon_b + \sum_{\substack{i,j \in C \\ j>i}} \varepsilon_{ij} && && \text{(REV)} \\
& \sum_{i \in C} b_i w_i + \sum_{\substack{i,j \in C \\ j>i}} b_i b_j w_{ij} + \varepsilon_b \geq \sum_{i \in C} b'_i w_i + \sum_{\substack{i,j \in C \\ j>i}} b'_i b'_j w_{ij} && && \forall b, b' : r(b') = r(b) + 1 \\
& w_i + w_j + w_{ij} + \varepsilon_{ij} \geq 0 && && \forall i, j \in C, j > i \\
& w_i \in [0,1] && && \forall i \in C \\
& w_{ij} \geq -2 && && \forall i, j \in C, j > i \\
& \varepsilon_b, \varepsilon_{ij} \geq 0 && && \forall i, j \in C, j > i, b \in B
\end{aligned}
$$

We solved the linear program for a large number of student preferences and in our environment, and none of the problems was feasible. We report results for a sample of students in Appendix B. We had preferences ranking 4000 to 12000 bundles for the courses of the winter term. None of these settings could be solved with objective value zero, that is, the generated preference lists are not representable with a linear model with adjustment-terms used by Budish et al. (2017). Even if it was possible to find such a vector of course-level weights, it would probably be very difficult to parametrize by students. The way Budish et al. (2017) elicit preferences might be sufficient for settings, where students only are interested in a very small subset of groups of the courses. However, assuming that students are able to adjust weights for up to 50 groups per course is utopian.

Eliciting preferences for hundreds of packages is a challenging problem, but the quality of any mechanism for CAP depends crucially on this input. There will be differences in the type of decision support one can provide in various types of applications. However, it is typically important that the timely preferences for students are captured.

4. Results

In Section 2.2 we have summarized first-order design goals for assignment problems: strategy-proofness, fairness, and efficiency. Now we introduce second-order design goals and respective metrics allowing us to compare the assignments of BPS and FCFS empirically. Then we provide numeric results and summarize the outcomes of a survey we conducted after the matching.

4.1. Metrics

Apart from efficiency, fairness, and strategy-proofness, *popularity* was raised as a design goal. An assignment is called popular if there is no other assignment that is preferred by a majority of the agents. Popular deterministic assignments might not always exist, but popular random assignments exist and can be computed in polynomial time (Kavitha et al. 2011). However, Brandt et al. (2017) prove that popularity is incompatible with very weak notions of strategy-proofness and envy-freeness, but it is interesting to understand the popularity of BPS vs. BRSD. In our empirical evaluation we analyze whether BPS or FCFS are more popular. To measure popularity we first define the function $\phi_i(b,b') : B \times B \to \{\pm 1, 0\}$ associated with the preference relations:

$$\phi_i(b,b') = \begin{cases} +1 & \text{if } b \succ_i b' \\ -1 & \text{if } b' \succ_i b \\ 0 & \text{else} \end{cases} \tag{4}$$

Definition 4 (Popularity). A random assignment $p \in \Delta$ is *more popular* than an assignment q, denoted $p \blacktriangleright q$, if $pop(p,q) > 0$ with

$$pop(p,q) = \sum_{i \in S} \sum_{b,b' \in B} p_{ib} \cdot q_{ib'} \cdot \phi_i(b,b') \tag{5}$$

A random assignment p is *popular*, if $\nexists q \in \Delta : q \blacktriangleright p$.

Apart from popularity, the size and the average or median rank are of interest. The *size* of a matching simply describes the number of matched agents. The *average rank* is only meaningful in combination with the size of the matching, because a smaller matching could easily have a smaller average rank. We report the average rank, because it has been used as a metric to gauge the difference in welfare of matching algorithms in Budish et al. (2017) and Abdulkadiroğlu et al. (2009), two of the few experimental papers on matching mechanisms.

The *profile* contains more information as it compares how many students were (fractionally) assigned to their first choice, how many to their second choice, and so on. The profile of two matchings is not straightforward to compare. We want to compare multiple profiles based on a single metric, and decided to use a metric similar to the *Area under the Curve of a Receiver Operating Characteristic* in signal processing (Hanley and McNeil 1982) which was already used by Diebold and Bichler (2017). The *Area Under the Profile Curve Ratio* (AUPCR) is the ratio of the Area Under the Profile Curve (AUPC) and the total area (TA) and is scaled between 0 and 100%, where the AUPC describes the integral below the profile curve. The AUPCR up to a specific rank n is equal to the probability that a matching mechanism will match a randomly chosen student higher than his n-th preference.

Definition 5 (AUPCR (Diebold and Bichler 2017)). Let C be the possible courses with $c \in C$ and Q be the sum of all capacities, regarding the students $i \in S$ the AUPCR is defined as follows:

$$\begin{aligned} TA(M) &= |C| \cdot min\{|S|, Q\} \\ AUPC(M) &= \sum_{r=1}^{|C|} |\{(i,c) \in M \mid rank(i,c) \leq r\}| \\ AUPCR(M) &= \frac{AUPC(M)}{TA(M)} \end{aligned}$$

For the allocation of bundles we have to rewrite the definition of the AUPCR.

Lemma 1 **(AUPCR)**. *With R denoting the number of possible ranks and $b \in B$, the AUPCR can be rewritten as:*

$$AUPCR(M) = \frac{1}{R} \sum_{r=1}^{R} \frac{|\{(i,b) \in M \mid rank(i,b) \leq r\}|}{|S|}$$

Appendix A provides a proof. We have already introduced stochastic orders in Section 2.2. We use second order stochastic dominance to compare two rankings (Levy 1992).

4.2. Empirical Results

The first application from the summer term 2017 comprised 1415 students and 67 courses (see Table 2). Overall, we had a list of 5847 different bundles for the summer term. We simulated FCFS via BRSD on the preferences collected for the BPS. BPS is weakly strategy-proof and in such a large application it is fair to assume that students do not have sufficient information about the preferences of others. In the survey, we will see that a small proportion of the students reported that they deviated from truthful bidding and did not report some of their preferred time slots. However, taking the preferences for bundles of tutor groups elicited for the BPS allows for a comparison with BRSD. To compare the result of BPS and BRSD we actually would have to run the BRSD for all permutations of the students. Note that computing probabilities of alternatives in RSD explicitly is $\#P$-complete (Aziz et al. 2013). We ran BRSD 1000 to 1,000,000 times with the same preferences but random permutations of the order of students and derived estimates for the different metrics. Since these results are very close, one can assume, that 1Mio runs of BRSD generate a good approximation to the (real) induced random matching.

4.2.1. Popularity For the data from the summer and the winter term, BPS is more *popular* than BRSD(1000000). 636 students prefer BPS to FCFS, while 96 students prefer FCFS to BPS. 683 students are indifferent (see Table 1). A positive popularity score as described in Definition 4 means, that BPS is more popular than the BRSD outcome and the score for BPS is 2.74 for the summer term and 3.41 for the winter term (compared to BRSD(1000000)). For the data from the winter term 754 students prefer BPS to FCFS, while 120 students prefer FCFS to BPS. 862 students are indifferent (see Table 1). Table 1 summarizes popularity and stochastic dominance for the summer and the winter term. The syntax for the *SD-preference* is the number of students preferring (BPS|BRSD(x)). It shows that BPS is preferable to BRSD according to SD-preference.

Metric	BRSD(1000000)
popularity summer	2.73635
popularity winter	3.41499
SD-prefer summer	(636|96)
SD-prefer winter	(754|120)

Table 1 Popularity and stochastic dominance of BPS vs. BRSD

4.2.2. Rank and Size Table 2 reports that in terms of average rank, average size, and the probability of being matched to one of the first 100 ranks BPS achieves higher scores in the summer term. Only the AUPCR for BRSD(1000000) is slightly better than for BPS. The computation times were negligible for BRSD (0.007 seconds per run). BPS required 0.12 seconds computation time with additional 6 minutes for the lottery algorithm in the summer term. This shows that BPS is a practical technique even for large assignment problems.

Metric	BPS	BRSD(1000000)
exp. rank	2.20163	2.20835
exp. size	1086.58	1085.79
prob. match (top 100)	0.767901	0.767345
AUPCR	0.747419	0.750782
weak envy	0	381
strong envy	0	1064

Table 2 Summary statistics for the summer term 2017.

In the BPS outcome 72.735% of the students receive an assignment ranked in their top ten while in BRSD 72.637% receive such an outcome (see Table 3 for BPS and 4 for BRSD w. 1 mio. permutations of the students). Table 3 reports the probability of being matched to a particular rank and the AUPC in percentage for BPS, and Table 4 shows the rank profile for BRSD.

Rank	1	2	3	4	5	6	7	8	9	10
Prob match(%)	54.174	5.691	4.542	2.025	1.506	0.935	1.167	0.940	1.141	0.613
AUPC in (%)	54.174	59.865	64.407	66.432	67.938	68.874	70.041	70.981	72.122	72.735

Table 3 Rank profiles for BPS in summer term 2017.

Rank	1	2	3	4	5	6	7	8	9	10
Prob match(%)	53.973	5.725	4.538	2.053	1.529	0.931	1.181	0.948	1.150	0.610
AUPC in (%)	53.973	59.697	64.236	66.289	67.818	68.748	69.929	70.877	72.027	72.637

Table 4 Rank profile BRSD(1000000) in summer term 2017.

The second application in the winter term included 1736 students and 66 courses. Overall, we had a list of 20,845 different bundles for the winter term. Again, BPS achieved better results than BRSD in all metrics (see Table 5).

In the BPS outcome 89.047% of the students receive an assignment ranked in their top ten while in BRSD 88.891% receive such an outcome (see Table 6 for BPS and 7 for BRSD with 1 mio. permultations of the students). The computation times were again very low. BPS required 0.382 seconds, but the lottery algorithm around 30 minutes due to the higher number of bundles generated in the winter term.

Metric	BPS	BRSD(1000000)
exp rank	1.97372	1.97873
exp size	1603.01	1600.84
prob match (top 100)	0.922253	0.922142
AUPCR	0.889512	0.888058
weak envy	0	451
strong envy	0	1202

Table 5 Summary statistics for the winter term 2017/2018.

4.2.3. Envy Our experiments in the summer and the winter term confirm the theoretical result that BPS is (strongly) *envy-free*. BRSD is neither weakly nor strongly envy-free. In the summer term, 1064 students do not fulfill the envy-freeness condition (see Definition 3), from which 381 students do not even fulfill the weak envy-freeness condition (see BRSD(1000000) in Table 2). Similarly, for the winter term 1202 students do not SD-prefer their outcome over the outcomes of every other student, and 451 of those students even prefer an outcome of another student (see BRSD(1000000) in Table 5).

Rank	1	2	3	4	5	6	7	8	9	10
Prob match(%)	73.596	7.083	3.392	1.660	1.041	0.698	0.465	0.447	0.366	0.299
AUPC in (%)	73.596	80.678	84.070	85.730	86.772	87.470	87.935	88.381	88.747	89.047

Table 6 Rank profiles for BPS in winter term 2017/2018.

Rank	1	2	3	4	5	6	7	8	9	10
Prob match(%)	73.452	7.046	3.382	1.673	1.040	0.704	0.486	0.443	0.358	0.307
AUPC in (%)	73.452	80.497	83.879	85.553	86.593	87.297	87.783	88.226	88.584	88.891

Table 7 Rank profile BRSD(1000000) in winter term 2017/2018

4.3. The Lottery of the Summer Term Instance

We already have discussed, that we still have to decompose the solution of BPS into a lottery over integral solutions, to choose a deterministic allocation. This subsection presents exemplary with the Data from summer term 2017, how such a lottery is structured, and how significant the problem of overallocation is in practice.

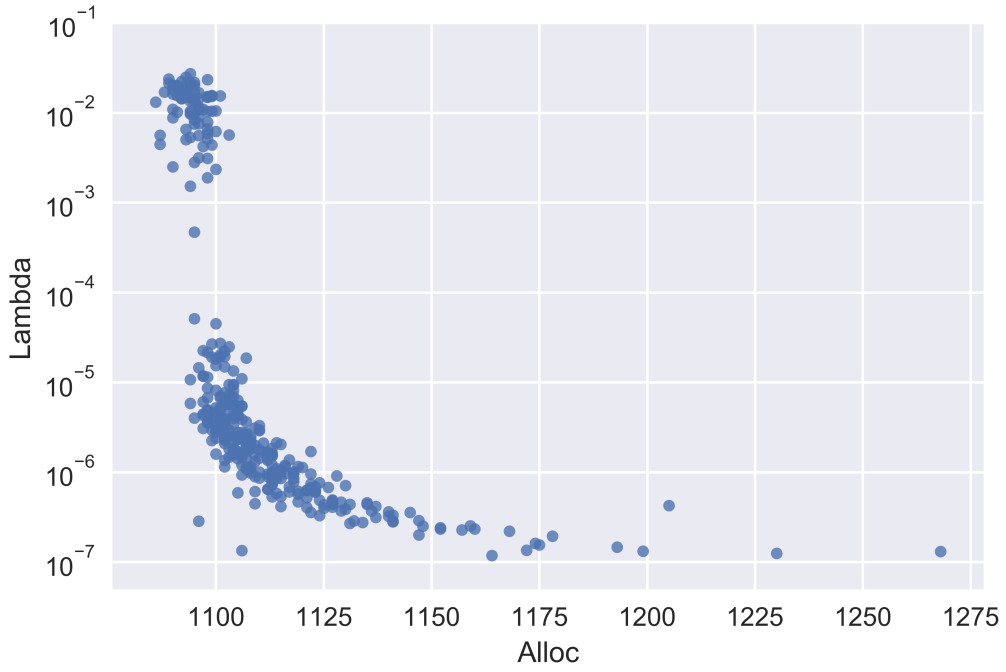

Figure 6 The lottery: Probabilities (λ) and size of the different deterministic matchings returned by Algorithm 3.

Figure 6 shows the lottery resulting from decomposing the BPS solution described in 2 into a lottery over approximative feasible integral solutions via Algorithm 3. We see that most solutions are close to the fractional solution in terms of number of allocated students (remember: the size of the BPS solution is 1086.58).

One interesting question is how the solutions with a bigger size differ from the matchings with a lower number of allocated students. We computed the average ranks of the deterministic solutions and compared them with the size of the particular matchings. Figure 7 shows the distribution of

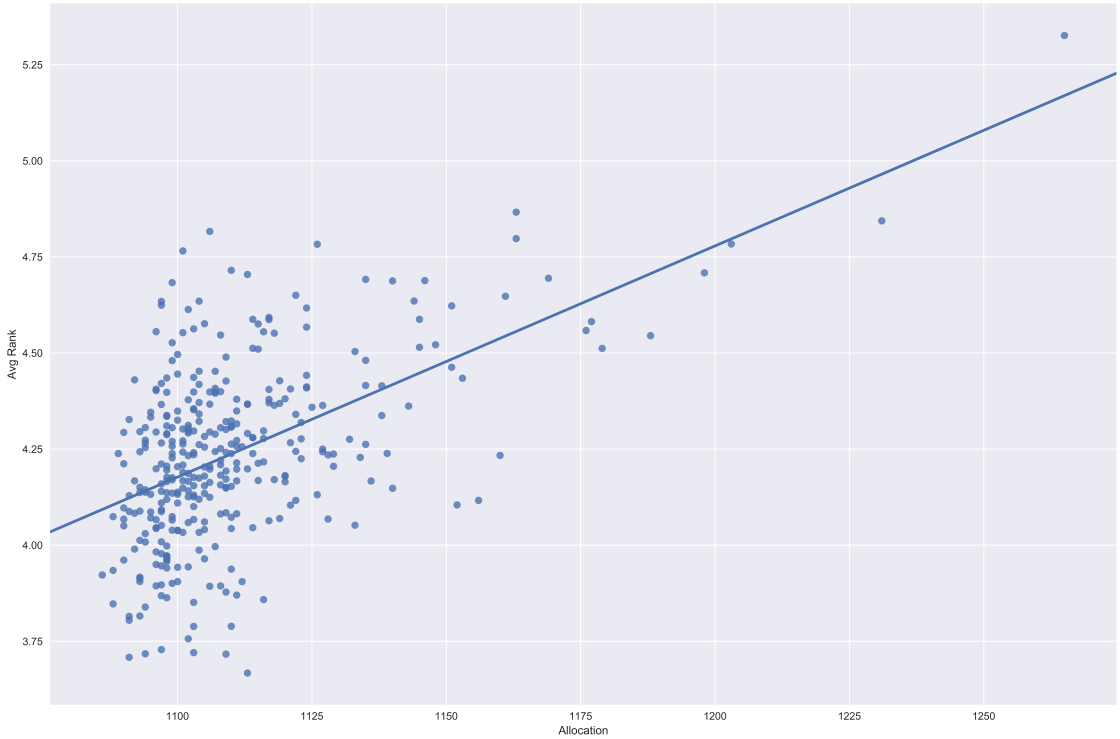

Figure 7 **Average rank vs. size of the matchings in the lottery.**

the different matchings in the lottery with respect to size and average rank. In the first fourth of the x-axis (the size of the matching) the variance is high for the average rank but the pattern gets clearer for a size higher than 1150 – there is a trade-off between size of the matching and the average rank of the allocated students.

In section 2.3 and 2.4 we already discussed that the capacity constraints of the courses might be violated. In the reported instance, $\ell = 4$. Hence, the worst case violation of these constraints is 3.

For the computation, we run the lottery with $\epsilon = 2.0$. Let $e_{x^{(k)},L}$ be the number of goods that experienced a supply violation by L units in the (integral) solution $x^{(k)}$, $\lambda^{(k)}$ the probability of matching $x^{(k)}$ and

$$E_L(Z) = \sum_{x^{(k)} \in Z} \lambda^{(k)} \cdot e_{x^{(k)},L}$$

shows us how often an over allocation of exact L seats occurs in the set of the lottery Z on average. With the settings mentioned above we receive:

- $E_1(X) = 5.36$
- $E_2(X) = 0.64$
- $E_3(X) = 0.07$

We see that an over allocation by 3 seats rarely happens. Even a violation of 2 seats occurs on average only in 0.64 of the 67 time courses. An over allocation of 1 seat occurs on average in 5.36

courses. In the actual application this overallocation did not even require special procedures and course organizers could typically accommodate one or two more students without problems.

The violations also barely change for results with $\epsilon = 1.0$:

- $E_1(X) = 5.23$
- $E_2(X) = 0.9$
- $E_3(X) = 0.06$

We informed the course organizer before the matching, that small violations of the capacities are possible and no one had a problem with that. If the capacities of some courses were tight, on could solve the problem, by defining a smaller capacity for those courses. Theoretically one had to reduce the capacity by $\ell - 1$ (3 in our case). However, our empirical results suggest, that a reduction of one seat should be sufficient, to ensure a feasible allocation with a high probability. The reduction of the capacities; however, come of cost of a lower efficiency of the matching in general.

4.4. Survey Results

After the students were assigned to the tutor groups and the courses started, we conducted a survey among the students using a 5-point Likert scale (1 = strongly agree, 2 = agree, 5 = strongly disagree). 169 students out of 1736 students participated in the survey in the winter term and we report their responses in Table 8. Note that the students were exposed to FCFS in other semesters and now participated in BPS, which allowed them to compare both mechanisms.

Students neither had to participate and we made clear that the feedback was used for research purposes only. The responses indicate that the majority of the students responding found the system easy to use and that they could express their preferences well. More than 50% agreed (2) or strongly agreed (1) to questions 1 to 6. A majority also considers the system as fair (question 7), but almost 22% of the respondents also disagreed to this statement. Note that students might have had an understanding of fairness that is different from envy-freeness or equal treatment of equals. For example, some students felt that in FCFS they could improve their assignment by making sure that they are among the first to register. This was perceived as fair as the additional effort would lead to higher chances of getting their best allocation as compared to those students who do not care about the assignment as much.

62.1% of the respondents were satisfied with the outcome (agreed or strongly agreed), while 28.4% were not. It is unclear how those students who did not respond perceived the outcome, but there is a tendency that students, who are unhappy with the outcome, rather respond than students who got a high ranked bundle. Hence, the sample of students who respond might be slightly biased towards unsatisfaction. The ranking and profile information reported earlier provides alternative information about satisfaction of students with the outcome.

	Question	1	2	3	4	5
1	I had no problems to select my time ranges in the weekly schedule	34.9	34.9	11.8	9.5	8.9
2	The ranking of the generated sets of time slots was easy	26.6	26.6	18.9	14.8	13.0
3	The instructions on the matching system were sufficient	25.4	37.3	18.3	10.1	8.9
4	The generated sets of tutorial groups met my expectations	37.9	27.8	10.1	9.5	14.8
5	I was able to express my preferences on sets of tutor groups well	42.6	24.9	13.6	7.7	11.2
6	I consider the way bundles are allocated through the matching system as fair	32.5	27.2	18.3	5.9	16.0
7	I am satisfied with the matching outcome	45.0	17.0	9.5	6.5	21.9
8	I felt like I had control over my schedule	29.0	18.9	13.0	17.2	21.9
9	I was expressing my preferences truthfully	72.4	13.4	4.2	3.6	6.5
10	I was strategically hiding some of my most preferred time slots	5.3	4.7	8.3	13.6	68.0
11	I was strategically hiding some of my least preferred time slots	16.0	12.4	16.0	12.4	43.2

Table 8 Survey results, values in %

85.8% of the respondents reported that they were expressing their preferences truthfully in BPS (agreed or strongly agreed), while around 10.1% did not (disagreed or strongly disagreed). 10.1% were also indicating that they were hiding some of their most preferred time slots, while even 28.4% agreed or strongly agreed to the statement that they were hiding some of their least preferred time slots. This high percentage needs to be seen in conjunction with the exponentially large set of possible packages. If a student provides many possible time slots, then the list of packages grows very large. Therefore, there might have been a tendency to narrow down the selection of acceptable time slots, i.e. not rank the least preferred time slots.

Still, the fact that a significant part of the students indicate that they did not report preferences truthfully is a tangible difference to FCFS. In FCFS, students only provide their single best package at the point in time, when they log in. This is simple, intuitive, and obviously strategy-proof. This property has to be traded off against the level of envy in BPS.

In a final question students were asked whether they prefer FCFS or BPS: 106 students (62,7%) preferred BPS, while 63 (37.3%) preferred FCFS. To understand the concerns of those students who preferred FCFS, it is useful to look at the written comments. Some students who provided comments were unhappy with the outcome, others were unhappy about the effort to rank-order their packages.

4.5. Discussion of Differences

The results from our field experiments and the survey reveal a number of interesting insights. Overall, BPS dominates BRSD on all metrics from our empirical evaluation in both field studies. It has a better average rank, a higher average size and a higher probability of matching, and it does not exhibit envy. However, the differences in average rank, average size, and the profile curve (AUPCR) are small, which is interesting given the fact that only a small number of preferences per student are considered via FCFS.

There are a number of reasons that help explain the close performance of BPS and FCFS in these metrics. First, Che and Kojima (2010) find that random serial dictatorship and probabilistic

serial become equivalent when the market becomes large, i.e. the random assignments in these mechanisms converge to each other as the number of copies of each object type grows, and the inefficiency of RSD becomes small. Our empirical results suggest that differences might also be small in large combinatorial assignment markets with limited complementarities.

Second, ordinal preferences do not allow to express the intensity of preferences. Suppose there are two students who both prefer course c_1 to c_2, each having one course seat only. No matter who gets course c_1, the average rank and size of the matching as well as the profile will be the same even though one student might desperately want to attend c_1, while the second student only has a mild preference for c_1. Without cardinal information about the intensity of a preference the differences in aggregate metrics can be small.

Third, an earlier comparison of FCFS with a deferred acceptance algorithm by Diebold et al. (2014) also showed that FCFS yielded surprisingly good results. While the average rank of FCFS was worse, the size of the matching resulting from FCFS was significantly larger compared to that from the deferred acceptance algorithm. For the combinatorial assignment problem, BPS actually had a larger average size than FCFS in both studies. For applications of matching in practice it is important to understand these trade-offs.

5. Conclusions

We report two large field studies and show that BPS performs well on a number of additional criteria including average rank, average size, probability of a matching among the first 100 ranks, and the overall profile of ranks (in terms of AUPC of a specific rank) assuming a complete, truthful, and strict ranking of all packages. The matching based on BPS is also more popular than BRSD based on the preferences submitted for BPS. The level of envy in FCFS is significant, even though the size of the packages that can be submitted is limited to the number of classes (three to four groups per package).

The assignment of tutor groups is specific as preferences are mainly about times of the week. The preferred time slots in a week are different from student to student. However, the way how tutor groups should be ordered within these time slots (e.g., time for breaks) can be described with a few parameters such that it was possible to generate packages according to a score. The feedback of students was that this automated ranking met their preferences well and we argue that this is a good way to address the missing bids problem in similar applications. In other applications, generating good bundles might not be as straightforward and this will have an impact on efficiency. Compact and domain-specific bid languages have been discussed in the auction literature (Bichler et al. 2011), and they could also be a possibility to allow mechanisms without transfers circumvent the missing bids problem.

The paper highlights basic trade-offs in market design without money: FCFS can be seen as a version of serial dictatorship which is ex post efficient, and obviously strategy-proof and treats students equally. It is also transparent and simple to implement and understand for students. BPS is a new randomized mechanism that is only weakly strategy-proof, but envy-free, and ordinally efficient, which is stronger than ex-post efficiency assuming strict preferences. Note that these properties hinge on the availability of strict preferences over all, exponentially many, bundles.

Even if the missing bids problem can be addressed, two important problems remain: First, in contrast to FCFS the BPS mechanism is not obviously strategy-proof and a part of the students in the survey already indicated that they either hid their most preferred or least preferred time slots strategically.[5] Second, the assumption of strict preferences is strong in the presence of exponentially many bundles. Unfortunately, extending PS or BPS to preferences with ties is not without loss. On the one hand, Katta and Sethuraman (2006) extended PS to preferences with indifferences and showed that it is not possible for any mechanism to find an envy-free, ordinally efficient assignment that satisfies even weak strategy-proofness as in the strict preference domain. On the other hand, with indifferences and random tie breaking efficiency cannot be guaranteed. Our preference elicitation technique generated a strict and complete ranking of course bundles based on a few input parameters and is one way to address these issues.

The key difference between BPS and FCFS is the absence of envy. The level of envy in FCFS is significant. Note that it might be even more pronounced if students were allowed to pick larger packages. Envy-freeness or stability has been raised as one of the arguments why the Gale-Shapley mechanism for simple assignment problems where agents have unit-demand (i.e. demand for only one course seat) is so successful in practice (Roth 2002). If the market outcome is unstable, there is an agent or apir of agents who have the incentive to circumvent the match. We argue that this property is as important for the assignment of course schedules. So, it envy-freeness matters, the elegant BPS mechanism has a number of attractive economic properties and is computationally tractable.

References

Abdulkadiroğlu, A., P. Pathak, A. Roth. 2009. Strategy-proofness versus efficiency in matching with indifferences: Redesigning the NYC high school match. *The American Economic Review* **88**(5) 1954–1978.

Abdulkadiroğlu, Atila, Parag Pathak, Alvin E Roth, Tayfun Sonmez. 2006. Changing the Boston school choice mechanism. Tech. rep., National Bureau of Economic Research.

Ashlagi, Itai, Mark Braverman, Avinatan Hassidim. 2014. Stability in large matching markets with complementarities. *Operations Research* **62**(4) 713–732.

[5] Remember that our empirical comparisons are based on the preferences reported in BPS. A part of these preferences might not have reflected the true preferences of participants, and the comparison might be biased towards BPS.

Ashlagi, Itai, Yannai A. Gonczarowski. 2015. Dating strategies are not obvious. *CoRR* **abs/1511.00452**.

Aziz, Haris, Felix Brandt, Markus Brill. 2013. The computational complexity of random serial dictatorship. *Economics Letters* **121**(3) 341–345.

Banker, Rajiv D, Robert J Kauffman. 2004. 50th anniversary article: The evolution of research on information systems: A fiftieth-year survey of the literature in management science. *Management Science* **50**(3) 281–298.

Bichler, M., J. Goeree, S. Mayer, P. Shabalin. 2014. Simple auctions for complex sales: Bid languages and spectrum auction design. *Telecommunications Policy* **38** 613–622.

Bichler, M., S. Schneider, K. Guler, M. Sayal. 2011. Compact bidding languages and supplier selection for markets with economies of scale and scope. *European Journal on Operational Research* **214** 67–77.

Birkhoff, Garrett. 1946. Three observations on linear algebra. *Univ. Nac. Tucumán. Revista A* **5** 147–151.

Bogomolnaia, Anna, Hervé Moulin. 2001a. A new solution to the random assignment problem. *Journal of Economic theory* **100**(2) 295–328.

Bogomolnaia, Anna, Hervé Moulin. 2001b. A new solution to the random assignment problem. *J. Economic Theory* **100**(2) 295–328. doi:10.1006/jeth.2000.2710. URL `https://doi.org/10.1006/jeth.2000.2710`.

Bowman, Edward H. 1963. Consistency and optimality in managerial decision making. *Management Science* **9**(2) 310–321.

Brandt, Felix. 2017. Rolling the dice: Recent results in probabilistic social choice. *Trends in Computational Social Choice* .

Brandt, Felix, Johannes Hofbauer, Martin Suderland. 2017. Majority graphs of assignment problems and properties of popular random assignments. *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 335–343.

Budish, Eric. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy* **119**(6) 1061–1103.

Budish, Eric, Gérard P. Cachon, Judd B. Kessler, Abraham Othman. 2017. Course match: A large-scale implementation of approximate competitive equilibrium from equal incomes for combinatorial allocation. *Operations Research* **65**(2) 314–336. doi:10.1287/opre.2016.1544. URL `https://doi.org/10.1287/opre.2016.1544`.

Budish, Eric, Judd Kessler. 2017. Can agents "report their types"? an experiment that changed the course allocation mechanism at wharton. *Chicago Booth Working Paper* .

Che, Yeon-Koo, Fuhito Kojima. 2010. Asymptotic equivalence of probabilistic serial and random priority mechanisms. *Econometrica* **78**(5) 1625–1672.

Diebold, Franz, Haris Aziz, Martin Bichler, Florian Matthes, Alexander Schneider. 2014. Course allocation via stable matching. *Business & Information Systems Engineering* **6**(2) 97–110.

Diebold, Franz, Martin Bichler. 2017. Matching with indifferences: A comparison of algorithms in the context of course allocation. *European Journal of Operational Research* **260**(1) 268–282.

Ehlers, L., B. Klaus. 2003. Coalitional strategy-proof and resource-monotonic solutions for multiple assignment problems. *Social Choice and Welfare* **21**(2) 265–280.

Fischer, Gregory W. 1972. Four methods for assessing multi-attribute utilities: An experimental validation. Tech. rep., MICHIGAN UNIV ANN ARBOR ENGINEERING PSYCHOLOGY LAB.

Gale, D., L. S. Shapley. 1962. College admissions and the stability of marriage. *American Mathematical Monthly* **69**(1) 9–15.

Gibbard, Allan. 1977. Manipulation of schemes that mix voting with chance. *Econometrica: Journal of the Econometric Society* 665–681.

Hanley, James A, Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology* **143**(1) 29–36.

Katta, Akshay-Kumar, Jay Sethuraman. 2006. A solution to the random assignment problem on the full preference domain. *Journal of Economic theory* **131**(1) 231–250.

Kavitha, Telikepalli, Julián Mestre, Meghana Nasre. 2011. Popular mixed matchings. *Theoretical Computer Science* **412**(24) 2679–2690.

Krishna, Aradhna, M Utku Ünver. 2008. Research noteimproving the efficiency of course bidding at business schools: Field and laboratory studies. *Marketing Science* **27**(2) 262–282.

Lavi, R., C. Swamy. 2011. Truthful and near-optimal mechanism design via linear programming. *Journal of the ACM (JACM)* **58**(6) 25.

Lee, Young Eun, Izak Benbasat. 2011. Research notethe influence of trade-off difficulty caused by preference elicitation methods on user acceptance of recommendation agents across loss and gain conditions. *Information Systems Research* **22**(4) 867–884.

Levy, Haim. 1992. Stochastic dominance and expected utility: survey and analysis. *Management Science* **38**(4) 555–593.

Li, Shengwu. 2017. Obviously strategy-proof mechanisms. *American Economic Review* **107**(11) 3257–87.

Mas-Colell, Andreu, Michael Dennis Whinston, Jerry R Green, et al. 1995. *Microeconomic theory*, vol. 1. Oxford university press New York.

Milgrom, P. 2010. Simplified mechanisms with applications to sponsored search and package auctions. *Games and Economic Behavior* **70**(1) 62–70.

Nguyen, T., A. Peivandi, R. Vohra. 2016. Assignment problems with complementarities. *Journal of Economic Theory* **165** 209 – 241.

Othman, Abraham, Christos Papadimitriou, Aviad Rubinstein. 2016. The complexity of fairness through equilibrium. *ACM Transactions on Economics and Computation (TEAC)* **4**(4) 20.

Pápai, S. 2001. Strategyproof and nonbossy multiple assignments. *Journal of Public Economic Theory* **3**(3) 257–271.

Pycia, Marek, Peter Troyan. 2016. Obvious dominance and random priority .

Roth, Alvin E. 1982. The economics of matching: Stability and incentives. *Mathematics of Operations Research* **7**(4) 617–628.

Roth, Alvin E. 2002. The economist as engineer: Game theory, experimentation, and computation as tools for design economics. *Econometrica* **40**(4) 13411378.

Santos, Brian L. Dos, Martin L. Bariff. 1988. A study of user interface aids for model-oriented decision support systems. *Management Science* **34**(4) 461–468.

Sönmez, Tayfun, M Utku Ünver. 2010. Course bidding at business schools. *International Economic Review* **51**(1) 99–123.

Von Neumann, John. 1953. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games* **2** 5–12.

Zhou, L. 1990. On a conjecture by gale about one-sided matching problems. *Journal of Economic Theory* **52**(1) 123–135.

	student 1	student 2	student 3	student 4
min lunch time	45 min		0 min	
time ranges	8am to 6pm		10am to 6pm	
feasible days (score)	Mo(2), Tu(4), We(5), Th(4), Fr(1)	Tu(5), We(5), Th(2)	Mo(5), Tu(3), We(5), Th(3)	Mo(5), Tu(5), We(5), Th(2), Fr(1)
# of bundles	8503	4120	4425	12370
$err(\varepsilon)$	≈ 0.005	≈ 0.004	≈ 0.004	≈ 0.005

Table 9 Parameters of the REV on the data for the winter term.

Appendix A: Proof

S ince students are interested in seats in more than one course, the sum of capacities of all selectable courses (tutor groups) is significantly higher than the number of participating students, therefore $min\,\{\,Q, |S|\,\} = |S|$

For matching problems with single unit demand, the number of possible ranks equals the number of courses, i.e. $|C| = R$. That is, we can rewrite $TA(M) = R \cdot |S|$. Since the students do not rank single courses but bundles of courses, we have to replace $c \in C$ by $b \in B$. We use this to get our conclusion:

$$AUPCR(M) = \frac{AUPC(M)}{TA(M)} = \frac{\sum_{r=1}^{R} |\{(i,b) \in M \mid rank(i,b) \leq r\}|}{R|S|}$$
$$= \frac{1}{R} \sum_{r=1}^{R} \frac{|\{(i,b) \in M \mid rank(i,b) \leq r\}|}{|S|}$$

Appendix B: Sample Preferences in REV

It is interesting to understand whether a simple course-level scoring rule as used in Budish et al. (2017) is expressive enough to describe the preference profiles with timely preferences in course assignment. Table 9 shows the parameters, the number of generated bundles as well as the respective objective function value of REV for the different sample preferences. None of the preference profiles we tested allowed for a feasible solution in REV.