Matching with Bundle Preferences: Tradeoff between Fairness and Truthfulness

Martin Bichler, Sören Merting[1], and Aykut Uzunoglu[2]

[1] Department of Informatics, Technical University of Munich, Germany
{bichler,soeren.merting}@in.tum.de
[2] Faculty of Business and Economics, University of Augsburg, Germany,
aykut.uzunoglu@wiwi.uni-augsburg.de

Abstract. Course assignment is a widespread problem in education. Often students have preferences for course schedules over the week. First-Come First-Served (FCFS) is the most widely used rule to assign students to courses in practice, but recent research led to alternatives with attractive properties. Bundled Probabilistic Serial (BPS) is a randomized mechanism satisfying ordinal efficiency, envy-freeness, weak strategy-proofness, and polynomial runtime. We report a first application of BPS in a large-scale course assignment application and discuss advantages over FCFS comparing a number of metrics such as the size, the average rank, the profile, and the popularity of the assignments. The exponential number of possible course schedules is a central problem in the implementation of combinatorial assignment mechanisms. We propose a new way to elicit preferences, which limits the number of parameters a student needs to provide. This yields a computationally very effective tool to solve course assignment problems with thousands of students in practice.

Keywords: Course Allocation, Combinatorial Assignment Problem, Randomization, Field Study

1 Introduction

Course assignment is arguably one of the most widespread assignment problems where money cannot be used to allocate scarce resources. Those problems of assigning students to different courses or whole schedules of courses 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 [1, 2] or course bidding [3, 4] 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) [5]. Similar problems arise when siblings should be

14th International Conference on Wirtschaftsinformatik,
February 24-27, 2019, Siegen, Germany

assigned to the same schools in school choice [6], or couples in the context of the hospital residency matching [7]. 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 [8]. 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 (TUM) that we will discuss. In the initial three semesters, 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 and they have timely preferences over course schedules that need to be considered, which makes it a combinatorial assignment problem. These problems are widespread 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 [5]. 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. 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, Cachon, Kessler and Othman [9] reports the empirical results at the Wharton School of Business. In addition, Budish and Kessler [10] 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 [11]. 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) [12-14]. 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, Peivandi and Vohra [15] 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 [16] called Bundled Probabilistic Serial (BPS). Nguyen, Peivandi and Vohra [15] 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 Contribution

We report on a field study and address issues 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. Theoretical contributions of assignment mechanisms largely focus on envy-freeness and efficiency as primary design desiderata. We also report on properties such as their size, their average rank, the probability of matching, the profile, and the popularity. These properties often matter in the choice of mechanisms beyond traditional ways to measure fairness and efficiency. For market designers it is important to understand the trade-offs.

Overall, we report on the assignment of 1415 students in the summer term 2017 to 67 tutor groups for four classes and the assignment of 1736 students in the winter term 2017/2018 to 66 tutor groups for four classes at the TUM using BPS.[1] 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.

Finally, 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

2 The Combinatorial Assignment Problem

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

[1] Not all students submitted a non-empty preference list. Therefore, we consider in our evaluation not all of the participating students (1439 in summer term, 1778 in winter term).

2.1 The Problem

Assigning objects to agents with preferences but without money is a fundamental problem referred to as *assignment problem* 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* $q = (q_1, q_2, \dots, q_m)$.

In the *combinatorial assignment problem* in the context of course allocation, every student $i \in S$ has a complete and transitive preference relation over subsets (or bundles) of elements of C. A preference profile $\succcurlyeq \in \wp^{|S|}$ is an n-tuple of preference relations.

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 $x_{ib} \in \{0,1\}$ 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.

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

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

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. As a result, the possible size of a bundle b is $size(b) \leq \ell \ll m$.

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 [17, 18] says that 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. However, the Birkhoff-von-Neumann theorem fails when students submit preferences for bundles of course seats. Nguyen, Peivandi and Vohra [15] 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 First-Order 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. More formally, a student $i \in S$ SD-prefers an assignment $p \in \Delta$ over $q \in \Delta$, $p \succcurlyeq_i^{SD} q$, if $\sum_{b' \succcurlyeq_i b} p_{ib'} \geq \sum_{b' \succcurlyeq_i b} q_{ib'}, \forall b \in B$. 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 $\succcurlyeq^{SD}$ is not a complete relation. That is there might be assignments p and q, which are not comparable with this relation.

One desirable property of matchings is *(Pareto) efficiency* such that no student can be made better off without making any other student worse off. That is, 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. $\nexists q \in \Delta: \forall i \in S: q \succcurlyeq_i^{SD} p$ 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 [16].

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. A random assignment $p \in \Delta$ is *(strongly) SD-envy-free*, if $\forall i, j \in S: p_i \succcurlyeq_i^{SD} p_j$. 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 $\psi: \wp^{|S|} \rightarrow \Delta$ that returns a random matching $p \in \Delta$. The mechanism $\psi(\succcurlyeq) = p$ 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 $\succcurlyeq_i = \succcurlyeq_j$ 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 his truthful preferences, no matter which preferences the other students report. A random assignment mechanism is (strongly) SD-strategy-proof if for every preference profile $\succcurlyeq$, and for all $i \in S$ and $\succcurlyeq_i'$ we have $\psi(\succcurlyeq_i, \succcurlyeq_{-i}) \succcurlyeq_i^{SD} \psi(\succcurlyeq_i', \succcurlyeq_{-i})$.

A random assignment rule ψ is *weakly SD*-strategy-proof if for every preference profile $\succcurlyeq$, there exists no $\succcurlyeq'$ *for some* student $i \in S$ such that $\psi(\succcurlyeq_i', \succcurlyeq_{-i}) \succcurlyeq_i^{SD} \psi(\succcurlyeq_i, \succcurlyeq_{-i})$. We will omit the prefix SD for brevity in the following. 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 [19].

Definition 1: OSP [19]. A strategy σ is obviously dominant if, for all other strategies σ', at any earliest information set where σ and σ' diverge, the best possible outcome from σ' is no better than the worst possible outcome from σ. A mechanism is obviously strategy-proof (OSP) if it has an equilibrium in obviously dominant strategies.

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 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 [20] showed that random dictatorship is the only anonymous and symmetric, strongly SD-strategy-proof, and ex post efficient assignment rule when preferences are strict. Pycia and Troyan [21] prove that RSD is a unique mechanism that is obviously strategy-proof, efficient, and symmetric in mechanisms without transfers. However, RSD is not always ordinally efficient, only ex post efficient [16]. Zhou [22] actually showed that no random mechanism for assigning objects to agents could 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) [16] 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, Peivandi and Vohra [15] is a generalization of PS to the combinatorial assignment problem and computes a fractional solution to (Demand) and (Supply). The BPS mechanism is also ordinally efficient, envy-free, and weakly strategy-proof if preferences are strict.

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. After a fractional solution x^* was found via BPS, it is implemented as a lottery over integral matchings satisfying the (Demand) and the relaxed (Supply) constraints, as described in [15].

3 Preference Elicitation

The Department of Informatics has been using stable matching mechanisms for the assignment of students to courses since 2014 [2]. 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. 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.[2]

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.

Budish, Cachon, Kessler and Othman [9] describe 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. Afterwards courses were 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.

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

However, our course allocation problem has a more special structure such that we can allow preferences that are more complex without asking for different weights for the courses. 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.

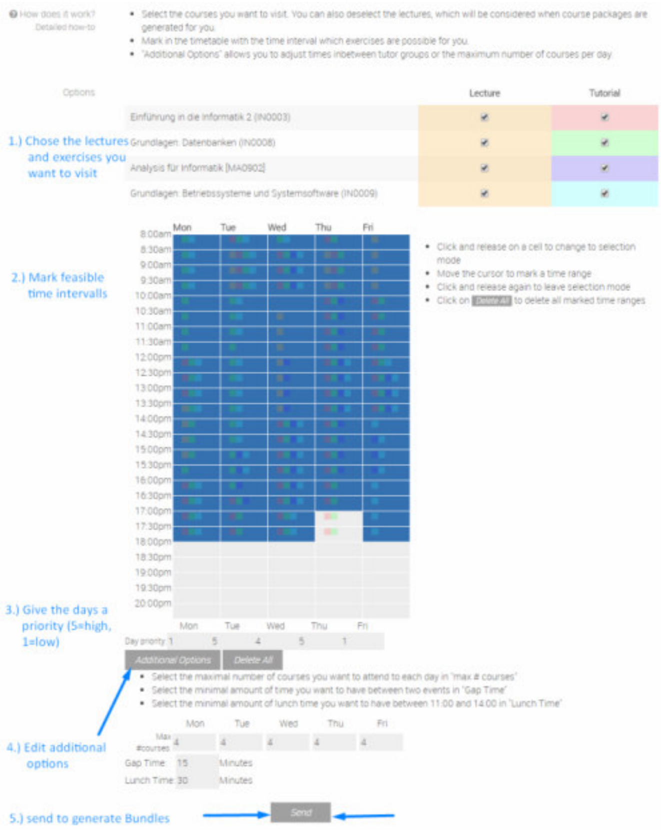
Students' preferences mainly concern their commute and the possibility to free large contiguous blocks of time (e.g., a day or a half-day) that they can plan for other activities (e.g., a part-time job). In larger cities such as Munich, the time that students spend for commuting is significant. Also 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.

First, 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 schedule*. 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 main web page and the main steps a student had to take are summarized in Figure 1.

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.

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 2).



How does it work?
Detailed how-to

- Select the courses you want to visit. You can also deselect the lectures, which will be considered when course packages are generated for you.
- Mark in the timetable with the time interval which exercises are possible for you.
- "Additional Options" allows you to adjust times inbetween tutor groups or the maximum number of courses per day.

1.) Chose the lectures and exercises you want to visit

Options	Lecture	Tutorial
Einführung in die Informatik 2 (IN0003)	☑	☑
Grundlagen: Datenbanken (IN0008)	☑	☑
Analysis für Informatik [MA0902]	☑	☑
Grundlagen: Betriebssysteme und Systemsoftware (IN0009)	☑	☑

2.) Mark feasible time intervalls

- Click and release on a cell to change to selection mode
- Move the cursor to mark a time range
- Click and release again to leave selection mode
- Click on Delete All to delete all marked time ranges

3.) Give the days a priority (5=high, 1=low)

	Mon	Tue	Wed	Thu	Fri
Day priority	1	5	4	5	1

4.) Edit additional options

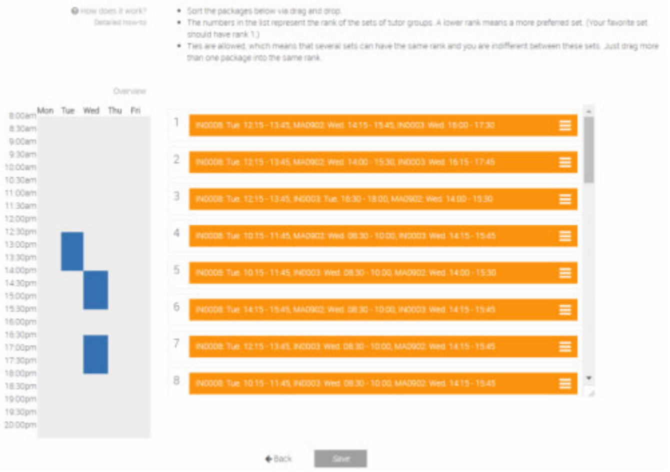
- Select the maximal number of courses you want to attend to each day in "max # courses"
- Select the minimal amount of time you want to have between two events in "Gap Time"
- Select the minimal amount of lunch time you want to have between 11:00 and 14:00 in "Lunch Time"

	Mon	Tue	Wed	Thu	Fri
Max #courses	4	4	4	4	4

Gap Time: 15 Minutes
Lunch Time: 30 Minutes

5.) send to generate Bundles

Figure 1. Process to rank-order packages



How does it work?
Detailed how-to

- Sort the packages below via drag and drop.
- The numbers in the list represent the rank of the sets of tutor groups. A lower rank means a more preferred set. (Your favorite set should have rank 1.)
- Ties are allowed, which means that several sets can have the same rank and you are indifferent between these sets. Just drag more than one package into the same rank.

Overview

1 IN0008: Tue: 12:15 - 13:45, MA0902: Wed: 14:15 - 15:45, IN0003: Wed: 16:00 - 17:30
2 IN0008: Tue: 12:15 - 13:45, MA0902: Wed: 14:00 - 15:30, IN0003: Wed: 16:15 - 17:45
3 IN0008: Tue: 12:15 - 13:45, IN0003: Tue: 16:30 - 18:00, MA0902: Wed: 14:00 - 15:30
4 IN0008: Tue: 10:15 - 11:45, MA0902: Wed: 08:30 - 10:00, IN0003: Wed: 14:15 - 15:45
5 IN0008: Tue: 10:15 - 11:45, IN0003: Wed: 08:30 - 10:00, MA0902: Wed: 14:00 - 15:30
6 IN0008: Tue: 14:15 - 15:45, MA0902: Wed: 08:30 - 10:00, IN0003: Wed: 14:15 - 15:45
7 IN0008: Tue: 12:15 - 13:45, IN0003: Wed: 08:30 - 10:00, MA0902: Wed: 14:15 - 15:45
8 IN0008: Tue: 10:15 - 11:45, IN0003: Wed: 08:30 - 10:00, MA0902: Wed: 14:15 - 15:45

Figure 2. Page with top-ranked packages

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[3].

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 and provide numeric results of our two matching instances.

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 [23]. However, Brandt, Hofbauer and Suderland [24] 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 $\phi_i(b, b'): B \times B \rightarrow \{\pm 1, 0\}$ associated with the preference relations, where $\phi_i(b, b') = 1$ if $b \succ_i b'$, -1 if $b' \succ_i b$ and 0 in any other case.

Definition 2: Popularity. A random assignment $p \in \Delta$ is *more popular* than an assignment q, denoted $p \rhd q$, if $pop(p, q) > 0$ with $pop(p, q) = \sum_{i \in S} \sum_{b,b' \in B} p_{ib} \cdot q_{ib'} \cdot \phi_i(b, b')$. A random assignment p is *popular*, if $\nexists q \in \Delta$: $q \rhd p$.

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 [9] and [25], 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.

[3] 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 activities required and could be confirmed without much effort.

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 [26], which was already used by [27]. With R denoting the number of possible ranks and $b \in B$, the *Area Under the Profile Curve Ratio (AUPCR)* for matching M can be defined as:

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

4.2 Empirical Results

Due to space constraints, we only analyze the results for the matching in winter term 2017/2018. This application comprised 1736 students and 66 courses (see Table 1). Overall, we had a list of 20,845 different bundles. 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, which would be necessary to strategically misreport their preferences. 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 [28]. 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. These estimates are close (see Table 1).

For our data, BPS is more popular than BRSD(1000000). 754 students prefer BPS to FCFS, while 120 students prefer FCFS to BPS (see Table 2). A positive popularity score as described in Definition 2 means, that BPS is more popular than the BRSD outcome and the score for BPS is 3.41 (compared to BRSD(1000000)).

Table 1. Summary statistics for the winter term 2017/2018.

Metric	*BPS*	*BRSD(1000)*	*BRSD(1000000)*
exp rank	1.97372	1.9784	1.97873
exp size	1603.01	1601.03	1600.84
prob match (top 100)	0.923394	0.922253	0.922142
AUPCR	0.889512	0.888184	0.888058
weak envy	0	427	451
strong envy	0	1050	1202

Table 2. Popularity and stochastic dominance of BPS vs. BRSD. The syntax for the SD-preference is the number of students preferring (BPS | BRSD(x)).

Metric	*BRSD(1000)*	*BRSD(1000000)*
popularity winter	1.93061	3.41499
SD-prefer winter	(690|299)	(754|120)

Table 1 reports that for all metrics BPS achieves better results. 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 3 for BPS and 4 for BRSD with 1 mio. permultations of the students). The computation times were negligible for BRSD (0.013 seconds per run). BPS required 0.382 seconds, but the lottery algorithm around 30 minutes due to the high number of bundles generated in the winter term.

Table 3. Rank profiles for BPS in winter term 2017/2018.

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 4. Rank profiles for BRSD(1000000)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

Our experiments confirm the theoretical result that BPS is (strongly) envy-free. BRSD is neither weakly nor strongly envy-free. In 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 1).

4.3 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 to explain the close performance of BPS and FCFS in these metrics. First, Che and Kojima [29] 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 expressing 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.

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 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 to understand for students. BPS is a new randomized mechanism that is only weakly strategy-proof, but envy-free, and ordinally efficient. 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[4]. 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 [30] 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 generates 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. If envy-freeness matters, the elegant BPS mechanism has a number of attractive properties and is computationally much less expensive compared to A-CEEI.

[4] 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.

References

1. Gale, D., Shapley, L.S.: College Admissions and the Stability of Marriage. American Mathematical Monthly 69, 9-15 (1962)
2. Diebold, F., Aziz, H., Bichler, M., Matthes, F., Schneider, A.: Course allocation via stable matching. Business & Information Systems Engineering 6, 97-110 (2014)
3. Krishna, A., Ünver, M.U.: Research Note - Improving the Efficiency of Course Bidding at Business Schools: Field and Laboratory Studies. Marketing Science 27, 262-282 (2008)
4. Sönmez, T., Ünver, M.U.: Course Bidding at Business Schools. International Economic Review 51, 99-123 (2010)
5. Budish, E.: The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. Journal of Political Economy 119, 1061-1103 (2011)
6. Abdulkadiroğlu, A., Pathak, P., Roth, A.E., Sönmez, T.: Changing the Boston School Choice Mechanism. National Bureau of Economic Research (2006)
7. Ashlagi, I., Braverman, M., Hassidim, A.: Stability in large matching markets with complementarities. Operations Research 62, 713-732 (2014)
8. Roth, A.E.: The Economics of Matching: Stability and Incentives. Mathematics of Operations Research 7, 617-628 (1982)
9. Budish, E., Cachon, G.P., Kessler, J.B., Othman, A.: Course Match: A Large-Scale Implementation of Approximate Competitive Equilibrium from Equal Incomes for Combinatorial Allocation. Operations Research 65, 314-336 (2017)
10. Budish, E., Kessler, J.: Can Agents "Report Their Types'"? An Experiment that Changed the Course Allocation Mechanism at Wharton. Chicago Booth Working Paper (2017)
11. Othman, A., Papadimitriou, C., Rubinstein, A.: The complexity of fairness through equilibrium. ACM Transactions on Economics and Computation (TEAC) 4, 20 (2016)
12. Milgrom, P.: Simplified Mechanisms with Applications to Sponsored Search and Package Auctions. Games and Economic Behavior 70, 62-70 (2010)
13. Bichler, M., Schneider, S., Guler, K., Sayal, M.: Compact bidding languages and supplier selection for markets with economies of scale and scope. European Journal on Operational Research 214, 67-77 (2011)
14. Bichler, M., Goeree, J., Mayer, S., Shabalin, P.: Simple Auctions For Complex Sales: Bid Languages and Spectrum Auction Design. Telecommunications Policy 38, 613-622 (2014)
15. Nguyen, T., Peivandi, A., Vohra, R.: Assignment Problems with Complementarities. Journal of Economic Theory 165, 209 - 241 (2016)
16. Bogomolnaia, A., Moulin, H.: A new solution to the random assignment problem. Journal of Economic theory 100, 295-328 (2001)
17. Birkhoff, G.: Three observations on linear algebra. Univ. Nac. Tucumán. Revista A 5, 147-151 (1946)
18. Von Neumann, J.: A certain zero-sum two-person game equivalent to the optimal assignment problem. Contributions to the Theory of Games 2, 5-12 (1953)
19. Li, S.: Obviously strategy-proof mechanisms. American Economic Review 107, 3257-3287 (2017)
20. Gibbard, A.: Manipulation of schemes that mix voting with chance. Econometrica: Journal of the Econometric Society 665-681 (1977)
21. Pycia, M., Troyan, P.: Obvious dominance and random priority. Working Paper (2018)
22. Zhou, L.: On a conjecture by gale about one-sided matching problems. Journal of Economic Theory 52, 123-135 (1990)
23. Kavitha, T., Mestre, J., Nasre, M.: Popular mixed matchings. Theoretical Computer Science 412, 2679-2690 (2011)

24. Brandt, F., Hofbauer, J., Suderland, M.: Majority graphs of assignment problems and properties of popular random assignments. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 335-343. (2017)
25. Abdulkadiroğlu, A., Pathak, P., Roth, A.: Strategy-proofness versus Efficiency in Matching with Indifferences: Redesigning the NYC High School Match. The American Economic Review 88, 1954-1978 (2009)
26. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143, 29-36 (1982)
27. Diebold, F., Bichler, M.: Matching with indifferences: A comparison of algorithms in the context of course allocation. European Journal of Operational Research 260, 268-282 (2017)
28. Aziz, H., Brandt, F., Brill, M.: The computational complexity of random serial dictatorship. Economics Letters 121, 341-345 (2013)
29. Che, Y.-K., Kojima, F.: Asymptotic equivalence of probabilistic serial and random priority mechanisms. Econometrica 78, 1625-1672 (2010)
30. Katta, A.-K., Sethuraman, J.: A solution to the random assignment problem on the full preference domain. Journal of Economic theory 131, 231-250 (2006)