RAM: A Collection of Mechanisms for (Indivisible) Resource Allocation in oTree [⋆]

Abstract

The growing theoretical literature on mechanism design and especially on the allocation of indivisible resources has created the need for experimental testing of mechanisms that evolved from matching theory in order to investigate the mechanisms' theoretical properties in practice. The collection of mechanisms offered by this package includes *Deferred Acceptance*, Gale's *Top Trading Cycles* and the *Boston Mechanism* in the context of School Choice Problems and the *University of Michigan Bidding System*, the *Gale-Shapley Pareto-Dominant Market Mechanism*, and *Random Serial Dictatorship* in the context of multi-unit resource allocation. All standalone apps are easy to use, customizable for many purposes, and are implemented in the *oTree* framework.

Keywords: Experimental Economics, Matching, oTree, Economic Software

1. Introduction

Since David Gale and Lloyd Shapley published their seminal paper introducing their Deferred Acceptance mechanism (Gale and Shapley, 1962) to solve the stable marriage problem, a whole strain of literature about allocation problems and their respective solution concepts was started. One prominent application of matching and mechanism design is the U.S. kidney exchange (pioneered by and implemented with the advise of Alvin Roth and co-authors (2004, 2005a, 2005b)). In the literature on the education sector, the debate about how to allocate students to schools (or colleges) was initiated by Milton Friedman (1955). Friedman's work was followed by many theoretical thoughts on mechanisms and their desired properties.[1] The central concepts will be introduced later in this Section.
Particularly in the last two decades, also the literature on experimental economics has started to pick up on some of the most important classes of allocation problems including school choice and multi-unit resource allocation. This paper contributes to the recent developements in school choice and multi-unit resource allocation problems and seeks to implement a software package to facilitate experimental hypothesis testing in the lab and

[⋆]The application referred to in this paper is always work-in-progress. For a live demo of the current version visit https://ram-demo.herokuapp.com/. For a full download version of the application during the revision process visit https://github.com/bpi87/ram.

[1]Sönmez and Ünver (2009) give a broad overview of the field of allocation problems and the properties of the respective most prominent solution algorithms, while Abdulkadiroglu and Sönmez (2003) discuss mechanism design in the school choice context.

the field. The motivation behind the experimental testing of allocation mechanisms is that the desired theoretical properties of the prevalent mechanisms do not necessarily hold true in practical applications, because cognitive abilities of agents in the mechanisms may be limited. Li (2017) provides a concept to describe and formalize these limitations and redefines mechanism design theory on a behavioral basis.

The difference between the two types of matching problems discussed in this paper is that in a multi-unit resource allocation problem, priorities of resources over agents are not given, i.e. resources are only consumable objects with no active role in the allocation problem. Furthermore, we assume that in the class of multi-unit resource allocation problems, resources can be matched with more than one agent and vice versa. Intuitively, the latter is not the case for school choice problems.
With the introduction of the *oTree* framework (Chen et al., 2016), the possibilities of experimental economists were greatly enriched. The *oTree* framework offers a more modern user interface thanks to it's state-of-the-art HTML5 front end. It utilizes the Python programming language at the back end, allowing experimenters to implement their work in a very elegant and flexible way of programming in a short time frame. The programming code can be enriched by *javascript* and other modular programming languages as well. Moreover, taking the lab to the field has never been easier, since *oTree* also runs on a web server such that the session can be accessed by any mobile device that is connected to the internet.
The remainder of the paper is structured as follows. In Section 2, the relevant problems and concepts are briefly explained. In the first part of Section 3, a short introduction on how to customize and set up an experiment with the software package at hand is provided, then I briefly introduce the implemented mechanisms, their theoretical properties and (dis-)advantages. In Section 4 I explain the key features and variables of the package and briefly address the limitations of the provided applications.

2. Allocation problems and their properties

In this paper, I consider the resource allocation problems defined in this Section:

Definition 1 (School Choice Problem). A *school choice problem* $S = (A, R, \succ_A, q, \succ_R)$ is defined as an allocation problem with

- a set of agents $A = \{a_1, a_2, \ldots, a_m\}$,
- a set of resources $R = \{r_1, r_2, \ldots, r_n\}$,
- a strict preference profile of the agents $\succ_A = (\succ_a)_{a \in A}$ over R,
- a quota $q = (q_r)_{r \in R}$ for each resource with $q_r \in \mathbb{Z}_{++}$,
- and a complete and transitive priority profile of resources over agents $\succ_R = (\succ_r)_{r \in R}$ over A.

There is a number of allocation algorithms for this class of problems in the literature, the most prominent being the *School Choice Top Trading Cycles* (TTC) mechanism by Abdulkadiroglu and Sönmez (2003), a modification of the *(Student Proposing) Deferred Acceptance* (DA) mechanism by Gale and Shapley (1962) and the *Boston School Choice Mechanism* (BOS).

Definition 2 (Multi-Unit Resource Allocation Problem). A *multi-unit resource allocation problem* $M = (A, R, \succ_A, q, t)$ is defined as an allocation problem with

- a set of agents $A = \{a_1, a_2, \dots, a_m\}$,
- a set of resources $R = \{r_1, r_2, \dots, r_n\}$,
- a strict preference profile of the agents $\succ_A = (\succ_a)_{a \in A}$ over R,
- a quota $q = (q_r)_{r \in R}$ for each resource with $q_r \in \mathbb{Z}_{++}$,
- and a timetable length $t = (t_a)_{a \in A}$ for each agent with $t_a \in \mathbb{Z}_{++}$.

Also for this class of problems, there are allocation algorithms that inherit some desirable properties, the most prominent being *Random Serial Dictatorship* (RSD). Besides RSD, and especially when it comes to an important application of this class of problems - course seat allocation at universities - the trivial *First Come, First Serve* system is often applied to allocate resources. In particular for the course allocation example, there are two more rather prominent mechanisms applied in the field: The *University of Michigan Bidding System* (UMBS) and it's extension by Sönmez and Ünver (2010a), the *Gale Shapley Pareto Dominant Market Mechanism* (GSPDM). The latter two mechanisms do not solely rely on preferences of agents, but (also) on a fictional currency units that inherit a cardinal component to the preferences.

We denote the outcome of an allocation problem as a *matching* $\mu : A \times R \to A \times R$ and a procedure that selects a matching for a specific problem as a *mechanism* $\phi : S \to \mu$ or $\phi : M \to \mu$. In mathematical terms, a matching is a function that assigns resources from a subset of resources to every agent ($\mu(a) = r, \mu(r) = a$). The most prominent criteria in the literature that are used to measure the quality of a matching mechanism are *strategy-proofness*, *stability* and *Pareto-efficiency.*

Definition 3 (Strategy-Proofness). A mechanism ϕ is strategy-proof if for any problem S or M, any agent $a \in A$ and any preference profile $\succ_a^*$

$$\phi[\succ_a, \succ_{-a}](a) \succeq_a \phi[\succ_a^*, \succ_{-a}].$$

This means when agents reveal their preferences and the central planner implements a matching using ϕ according to the revealed preference profile, it is a weakly dominant strategy for any agent $a \in A$ to reveal his/her preference profile truthfully (Sönmez and Ünver, 2009).

Definition 4 (Stability)**.** An allocation of a mechanism ϕ is stable, if it eliminates justified envy, meaning that it can not be the case that $\emptyset \succeq_a \mu(a)$ for any $a \in A$ (no blocking individual) or $a \succeq_r \mu(r), r \succeq_a \mu(a)$ for any $a \in A, r \in R$ (no blocking pair).

Definition 5 (Pareto-efficiency)**.** A matching μ is **Pareto-efficient** if there exists no other matching ν in which all agents are weakly better off and (at least) one agent is strictly better. Formally, $\nu(a) \succeq_a \mu(a)$ for all agents $a \in A$ and $\nu(a) \succ_a \mu(a)$ for some agent $a \in A$.

3. Implemented Mechanisms and User Settings

In the RAM package, six commonly used mechanisms from the matching literature were picked and implemented in the *oTree* framework. This is supposed to support future research in the field of matching. All six mechanisms are standalone *oTree* apps which can be seamlessly combined with other *oTree* apps and are highly customizable by the experimenter. For this purpose, a file (`user_settings.py`) containing the *oTree* class `Constants` is placed in the application folder and allows the experimenter to customize appearance, mechanism parameters, and general settings for every mechanism. For a detailed description of the functionality of every customizable parameter, `user_settings.py` also includes explanations for all variables.[2] Furthermore, a user manual is attached in the Download section of this paper.

The customizable variables for the **design settings** of every mechanism include the number of agents per group (`players_ per_group`), an induced utility for each resource (`valuations_t*`, with `*` representing the type of the subset of agents. More on that feature later in this Section.), and a vector with the quotas for each resource (`capacities`).
Appearance settings include a switch to display an instructions[3] page and the option to include an example on how the mechanism works (`instructions`, `instructions_example`), and a switch to display a results page (`results`). Further, it is possible to add a second button on `Decision.html`, where subjects must confirm their input (`confirm_button`). This has proven to be a desirable feature, since subjects tend to accidentally hit the Enter-Key while making their decisions. All apps in the RAM package come with two different framing options. The experiment can either choose a neutral framing (Participants/Resources), or framed version (`application_framing`) of the experiment. The framed version represents the most common applications for the respective type of allocation problem (Participants/Schools or Students/Courses).
The third class of provided customizations are **information settings**. Since information is key to many experimental design, the user of this package has some convenient features to adjust information. If `show_types=True`, subjects will have a hint and on the decision page that valuations may differ within their group. If `show_valuations=True`, the other types'

[2]The outsourcing of the `Constants` class was originally seen in and is related to Holzmeister and Pfurtscheller (2016).

[3]The instructions and examples provided are partly taken from the seminal experimental papers cited in this paper and are only samples.

valuation profiles will be revealed to all subjects. Furthermore, there is a switch to display the quota of each resource (`show_capacities`) in the instructions and on `Decision.html`. Since experimental subjects submit their preferences (and bid their points) based on an induced utility profile, it is desirable that this utility profile can vary across subjects (i.e. that different subjects can be incentivized to submit different preference profiles). This feature is implemented within the `valuations_t*` variable(s), where the experimenter can specify multiple valuation profiles. I.e., it is possible to specify as many valuation profiles as `players_per_group`, with the restriction that the number of valuation profiles must be completely divisible by `players_per_group`. E.g., if `players_per_group=6`, there can be 1, 2, 3, or 6 different valuation profiles. The program code handles the valuation profiles as lists and translates them into types using the built-in method `role(self)`. Only if multiple valuation profiles are specified, the switches `show_types` and `show_valuations` can be used to control the information of agents in the experiment.
The variables with special functionality for different mechanisms will be explained in the respective Subsections of this Section.

3.1. School Choice Mechanisms

3.1.1. Student Proposing Deferred Acceptance Mechanism

The student proposing DA mechanism is probably the best known mechanism in the school choice literature. It is a school choice compatible refinement for the originally introduced DA mechanism by Gale and Shapley (1962). This mechanism was prominently used in Hong Kong college seat assignment and is also used in the U.S. to assign interns to hospitals (Roth and Peranson, 1997; Peranson and Roth, 1999).

Algorithm 1 (Student Proposing Deferred Acceptance Mechanism)**.** *After collecting preferences of agents and priorities of resources, the algorithm proceeds as follows:*

Step 1: *Every agent applies to the resource stated as her top preference. Each resource then reviews all applications of agents and "tentatively" assigns the agents to it in the order of its' priority list until either its' quota is depleted or there are no applications left and rejects all other agents.*

$\vdots$

Step l: *Every agent that was rejected in the previous step applies to the resource stated as her l^{th} preference. Each resource then reviews all applications of agents in step l and the "tentative" assignment of step $(l-1)$ and "tentatively" assigns the agents to it in the order of its' priority list until either its' quota is depleted or there are no applications left. The algorithm terminates after no application is rejected in a step or after there are no applications left to consider in the agents' preference lists.*

The DA mechanism has some desirable features:

Proposition 1 (Elimination of justified envy - DA)**.** *The Student Proposing Deferred Acceptance Mechanism eliminates justified envy in the context of school choice (Abdulkadiroglu and Sönmez, 2003).*

Proposition 2 (Pareto-dominance - DA)**.** *The Student Proposing Deferred Acceptance Mechanism Pareto-dominates all other mechanisms that do also eliminate justified envy (Gale and Shapley, 1962).*

Proposition 3 (Strategy-proofness - DA)**.** *The Student Proposing Deferred Acceptance Mechanism is strategy-proof (Dubins and Freedman, 1981; Roth, 1982).*

3.1.2. School Choice TTC Mechanism

Another common mechanism is based on Gale's Top Trading Cylces Mechanism and was adapted for this class of allocation problems (Abdulkadiroglu and Sönmez, 2003).

Algorithm 2 (School Choice TTC Mechanism)**.** *After collecting preferences of agents and priorities of resources, the algorithm proceeds as follows:*

Step 1: *Each resource points to its top ranked agent in the priority list and each agent points to her top ranked resource in her preference list. Whenever a pointer cycle is found, all agents in the cycle are assigned to the resources they point to. The quotas of all resources in the cycle are reduced by one. All agents that were assigned in this step and all resources with depleted quota are removed.*

$\vdots$

Step l: *The algorithm continues the procedure with the remaining agents and resources. The algorithm terminates when no more cycles are found.*

The TTC mechanism inherits the following properties:

Proposition 4 (Pareto-efficiency - TTC)**.** *The TTC mechanism is Pareto-efficient (Abdulkadiroglu and Sönmez, 2003).*

Proposition 5 (Strategy-proofness - TTC)**.** *The TTC mechanism is strategy-proof (Abdulkadiroglu and Sönmez, 2003).*

Especially when comparing TTC to DA, the central conflict faced by theorists and policy-makers is the trade-off between the elimination of justified envy and Pareto-efficieny. However, there seems to be no clear consensus about which mechanism shall be chosen in the field (see Abdulkadiroglu and Sönmez (2003)).

3.1.3. Boston School Choice Mechanism

Compared to TTC and DA, the Boston School Choice mechanism is an inferior mechanism in theory and in practice. This is due to the lack of strategy-proofness and a lack of efficiency in practice. However, since the Boston Mechanism is still in place in practice and is perceived to be good by its participants, there still may be a need for practical testing in an experimental context. In the recent years, Kojima and Ünver (2014) and Hatfield et al. show that the Boston Mechanism also has some desirable features apart from classical matching theory.

Algorithm 3 (Boston Mechanism)**.** *After collecting preferences of agents and priorities of resources, the algorithm proceeds as follows:*

Step 1: *Every agent applies for the resource stated as her top preference. Each resource then reviews all applications of agents and assigns the agents to it in the order of its priority list until either its quota is depleted or there are no applications left. All agents that were assigned to a resource and all resources with depleted quota are removed.*

$\vdots$

Step l: *All remaining agents are considered in this step and apply for the resource stated as their l^{th} preference. Each remaining resource then reviews all applications of agents and assigns the agents to it in the order of its remaining priority list until either its quota is depleted or after there are no applications left.*

For all three mechanisms of this class, `user_settings.py` features an additional switch that allows the user to control the subjects' information concerning the resources' priorities: If `show_priorities=True`, subjects can see their priority ranking for all resources (i out of n) on the decision screen.

3.2. Multi-Unit Resource Allocation Mechanisms

A prominent practical example for this class of problems is the allocation of course seats to university students. Today's course allocation at business schools is mainly achieved either via serial dictatorship rules or via bidding-based mechanisms. However, the trend towards one or the other system is not really clear. I will now introduce three allocation mechanisms that are frequently used in practice.

3.2.1. Random Serial Dictatorship

A very common approach to efficiently allocate courses to students is the use of serial dictatorship rules (also frequently referred to as *priority mechanisms*), which has been studied at length in the economic literature (see e.g. Chen and Sönmez (2002) and Abdulkadiroglu and Sönmez (1999)). The implementation at hand is a simultaneous version of RSD, meaning that all agents make their decisions at the same time.[4] The mechanism works as follows:

Algorithm 4 (Random Serial Dictatorship). *After collecting preferences of agents, a random ordering of agents is obtained and the algorithm proceeds as follows:*

Step 1: *The first agent in the random ordering gets to choose all of her most preferred resources in her preference list and fills up her timetable starting from the highest ranked resource. The quotas of chosen resources by the first agent are reduced by one. All resources with depleted quota are removed.*

$\vdots$

Step l: *The l^{th} agent in the random ordering gets to choose all of her most preferred resources in her preference list with a non-depleted quota and fills up her timetable starting from the highest ranked resource. The algorithm terminates either after all resources are depleted or the preference lists of all agents have been considered.*

The implemented *oTree* app has the additional option (`show_position`) to show the position of the agent in the random ordering obtained by the algorithm on `Decision.html`. Random Serial Dictatorship possesses the following properties:[5]

Proposition 6 (Pareto-efficiency - RSD). *The RSD mechanism is Pareto-efficient (Pathak, 2011).*

Proposition 7 (Strategy-proofness - RSD). *The RSD Mechanism is strategy-proof (Pathak, 2011).*

3.2.2. University of Michigan Bidding System

The UMBS relies on point bidding only. This is a common approach in practice to allocate university course seats. It works as follows:

Algorithm 5 (University of Michigan Bidding System). *After all agents have made their decisions, all bids are ordered in a single list in a descending order (from the highest bid to the lowest bid). Then, each bid is considered - beginning with the first item in the list - separately. A bid is successful if*

- *the quota of the corresponding resource is not depleted, and*
- *the agent still has unfilled slots in her timetable.*

[4]A sequential version of RSD is a feature to be implemented in a forthcoming version of this package (Effective: December 3, 2018).

[5]Furthermore, sequential random serial dictatorship also fulfills the requirements to be obviously strategy-proof as defined in Li (2017).

If a bid is successful, the agent is allocated with the resource, the quotas and timetables of the corresponding resources and agents are reduced by 1, and the next bid is considered. If the bid is not successful, the agent is declined for this resource and the next bid will be considered. The algorithm terminates after all bids in the list were considered.

Rresearch on course bidding mechanisms suggest that a *conflicting dual role of bids* (Krishna and Ünver, 2008) in many college course bidding mechanisms can lead the mechanism in place to deliver inefficient results. The mechanism uses the bids to i) infer the preferences of the students for certain courses and ii) to obtain the claim of a student for certain a course (i.e., the preference intensities). In other words, the bids reflect preferences and beliefs about prices. According to Krishna and Ünver (2008), the UMBS mechanism suffers crucially from this problem. However, it is still in place at many universities.
The implemented *oTree* app features three additional variables for customization. First, it is possible to set the point endowment (`endowment`). However, the endowment has to be equal for all agents in the market. Second, the experimenter can enforce this endowment constraint to be binding (`enforce_binding`). I.e., if `enforce_binding=True`, agents have to use all their bidding points. Third, I implemented the possibility to show a javascript live counter such that subjects can see how many bidding points they have left to bid (`show_counter`).

3.2.3. Gale-Shapley Pareto-dominant Market Mechanism

In order to get rid of the inefficient outcomes from the UMBS, Sönmez and Ünver (2010b) proposed the Gale-Shapley Pareto-dominant Market Mechanism. This mechanism relies on asking agents not only for a vector of bids for desirable courses, but also for an ordinal ranking of desirable resources. The mechanism itself works like a synthesis of the student-proposing deferred acceptance algorithm and a bidding-based algorithm. Using this mechanism, Krishna and Ünver (2008) can resolve the issue of the conflicting dual role of bids, showing that the mechanism at least always outperforms the point bidding system in theory.

Algorithm 6 (Gale Shapley Pareto Dominant Market Mechanism)**.** *After collecting preferences and bids of agents and breaking ties for equal bids, the algorithm proceeds as follows:*

Step 1: *Each agent applies to his/her top resources based on her stated preferences. Each resource rejects all but the highest bidding agents up to its quota among those who have applied. Those who are not rejected are kept on hold.*

$\vdots$

Step l: *Each agent who is rejected from any resources in Step $(l-1)$ applies to his/her best remaining resources based on her stated preferences. Each resources considers the new applications together with the applications on hold and rejects all but the highest bidding agents. Those who are not rejected are kept on hold. The procedure terminates either when no application is rejected or after there are no applications left to consider in the agents' preference lists.*

Like for the UMBS mechanism, `user_settings.py` features three additional variables for customization. First, it is possible to set the point endowment (`endowment`). Second, the experimenter can enforce this endowment constraint to be binding (`enforce_binding`). Third, I implemented the possibility to show a javascript live counter such that subjects can see how many bidding points they have left to bid (`show_counter`).

4. Software Features, Variables, and Limitations

4.1. Admin Report

All modelled apps feature an overview of the key outcomes of the respective mechanism. Within the *oTree* framework, this feature is called *Admin Report*. From the *oTree* admin interface, the experimenter can click on the *Admin Report* tab in the top bar, which opens `AdminReport.html` in the browser window. On this page, the experimenter has a live observation tool of subjects' decisions during the experiment. For all apps, `AdminReport.html` first summarizes the parameterization of the experimental session including all key variables set in `user_settings.py`. Furthermore, it displays all decisions made by subjects during the experiment (submitted preferences and (in UMBS and GSPDM) submitted bids for all resources). Additionally, on `AdminReport.html` the experimenter has the possibility to export a compact `.xls` version of the relevant data suitable for most proposes. Here, also the successfully obtained resources for each agent are saved.[6]

4.2. Multi-Language-Support

The `i18n` internationalization package was used to translate all texts on the front-end provided in the apps of this package. As a sample, the RSD and UMBS apps are available in German language. However, all apps are ready for internationalization.

4.3. Variables

The variables saved to the database include all the relevant decisions (submitted preferences and/or bids) made by participants of the experimental sessions. Also, the outcomes of the particular markets are saved to the database. This includes binary variables that indicate if a participant was allocated to the resource(s), the market clearing bids (only in UMBS and GSPDM), and the subjects' payoffs. Moreover, the specified type as set by the experimenter in `user_settings.py` are saved to the database.

4.4. Limitations and Planned Features

The application is tested up to a group size of 10 and a number of resources of 10. Larger groups should in general be possible, but are not officially supported. Bootstrap is not officially supported, yet. However, all apps are tested with standard laptops, desktop PCs and medium-sized tablets. It is not recommended to use the apps on smaller devices.

[6]However, this compact dataset is not as rich as the dataset prepared by *oTree* itself. For full data export, the user can click on the *Data* tab in the top bar of the *oTree* admin interface.

A more sophisticated bootstrapping is a planned feature. The RAM package is constantly work-in-progress and will be updated and extended in the future. For bug reports and feature requests, please feel free to contact the author at any time.

References

Abdulkadiroglu, A., Sönmez, T., 1999. House allocation with existing tenants. Journal of Economic Theory 88, 233–260.

Abdulkadiroglu, A., Sönmez, T., 2003. School choice: A mechanism design approach. American Economic Review 93, 729–747.

Chen, D.L., Schonger, M., Wickens, C., 2016. otree — an open-source platform for laboratory, online, and field experiments. Journal of Behavioral and Experimental Finance 9, 88–97.

Chen, Y., Sönmez, T., 2002. Improving efficiency of on-campus housing: An experimental study. American Economic Review 95, 1669–1686.

Dubins, L.E., Freedman, D.A., 1981. Machiavelli and the gale-shapley algorithm. The American Mathematical Monthly 88, 485–494. doi:10.1080/00029890.1981.11995301.

Friedman, M., 1955. The role of government in education, in: Solo, R.A. (Ed.), Economics and the public interest. Rutgers University Press, New Brunswick, NJ, pp. 123–144.

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

Hatfield, J.W., Kojima, F., Narita, Y., .

Holzmeister, F., Pfurtscheller, A., 2016. otree: The "bomb" risk elicitation task. Journal of Behavioral and Experimental Finance 10, 105 – 108.

Kojima, F., Ünver, M.U., 2014. The "boston" school-choice mechanism: an axiomatic approach. Economic Theory 55, 515–544.

Krishna, A., Ünver, M.U., 2008. Improving the efficiency of course bidding at business schools: Field and laboratory studies. Marketing Science 27, 262–282.

Li, S., 2017. Obviously strategy-proof mechanisms. American Economic Review 107, 3257–3287.

Pathak, P., 2011. The mechanism design approach to student assignment. Annual Review of Economics 3, 513–536.

Peranson, E., Roth, A.E., 1999. The redesign of the matching market for american physicians: Some engineering aspects of economic design. American Economic Review 89, 748–780.

Roth, A.E., 1982. The economics of matching: stability and incentives. Mathematics of Operations Research 7, 617–628. doi:10.1287/moor.7.4.617.

Roth, A.E., Peranson, E., 1997. The effects of the change in the nrmp matching algorithm. national resident matching program. Journal of the American Medical Association 278, 729–732.

Roth, A.E., Sönmez, T., 2005. A kidney exchange clearinghouse in new england. American Economic Review 95, 376–380.

Roth, A.E., Sönmez, T., Ünver, M.U., 2004. Kidney exchange*. The Quarterly Journal of Economics 119, 457–488.

Roth, A.E., Sönmez, T., Ünver, M.U., 2005. Pairwise kidney exchange. Journal of Economic Theory 125, 151–188.

Sönmez, T., Ünver, M.U., 2009. Matching, Allocation, and Exchange of Discrete Resources. Boston College Working Papers in Economics 717. Boston College Department of Economics.

Sönmez, T., Ünver, M.U., 2010a. Course bidding at business schools. International Economic Review 51, 99–123.

Sönmez, T., Ünver, M.U., 2010b. Matching, allocation, and exchange of discrete resources. International Economic Review 51, 99–123. doi:10.1111/j.1468-2354.2009.00572.x.