Mechanism Design without Money II: House Allocation, Kidney Exchange, Stable Matching

ETH Zürich Peter Widmayer, Paul Dütting

Looking at the past few lectures you may have gained the impression that money not only makes the world go round, but also Mechanism Design. In fact, the existence of money was essential for most of our results. For instance, it allowed us to align the incentives of individual players with our greater goals such as maximizing social welfare.

On the other hand it is not difficult to think about strategic situations, where the use of money is either forbidden or considered immoral (or both). A concrete example consider the "market" for organ donations. Allowing people to pay for organs would be considered unfair as it would give richer people better access to replacement organs. Even worse, it would almost certainly lead to organ trade with all its undesirable repercussions.

So what can we do in situations like this? What can we achieve with mechanisms when we are not allowed to use money? Our focus will be on ways to exhibit truthful behavior while at the same time constructing good solutions to our problems. We will study this question in three exemplary situations. The allocation of resources such as houses, kidney exchange, and stable matchings or stable marriages.

1 House Allocation

In this resource allocation problem, each of n players has a house, and each player has a strict preference list (a total order) over all n houses. The lists describe individual preferences, i.e., they can vary arbitrarily across players.

The question now is whether one can rearrange the allocation of houses to players so that overall the players are better off. We assume that each player cares only about the house she gets, and not about who gets which house among the other players. We further assume that a mechanism cannot force players to accept a new house, but if the mechanism proposes to a player a house that is better for her (i.e., higher on her preference list) than what she currently owns, she will accept. It therefore makes no sense for a mechanism to propose to a player a house that is worse for her than her current house.

It is not hard to get an intuition for how players would behave in simple situations. For example, if a player's house is highest on her own preference list, she will not be willing to exchange it for any other house, but keep it under all circumstances (i.e., for all other players' preferences). On the other hand, if I like your house best and you mine, we will exchange houses. Note that we can interpret both these cases as cycles. The first situation corresponds to a cycle of length one, a so-called self loop, and the second situation to a cycle of length two. More generally, we can think of longer cycles along which we could swap houses. This suggests the following algorithm.

Top Trading Cycles Algorithm (TTCA)

0. Initially, all players and all houses remain.

   While players remain, do the following:

1. Let each remaining player point to her favorite (top) remaining house. This describes a directed graph where each vertex has out-degree one.

2. Reallocate as suggested by the directed cycles in the graph (including self loops), and delete the reallocated houses and players. All other players keep their current houses.

The TTC Algorithm has four important and highly desirable properties.

1. Termination. The algorithm terminates, because Step 2 always reallocates along at least one cycle. The reason for the existence of a cycle is the fact that each vertex has exactly one outgoing edge, and therefore we can start at any vertex and follow outgoing edges as long as we wish. At some point, a vertex must repeat, and a cycle is found.

2. Weakly improved allocation. At the end, each agent has a house that she likes at least as much as her initial house. The reason is that after Step 2 of TTCA, all remaining players keep their houses, and the agents that got a house in fact pointed to this house. If a player points to her own house in an iteration of the loop, she gets it. Other houses can be allocated to a player only earlier, when the player points to a house higher up on her list than her own house.

3. Incentive compatibility. TTCA is incentive compatible when preference lists are private. This is a key feature that a house allocation mechanism must have. The reason for incentive compatibility can be explained inductively. Each player who gets a house in the first iteration of the loop will get her first choice on the preference list, so she has no incentive to lie. Let N_i denote the set of players who get a house in iteration i. We have just seen that for each player in N_1, TTCA is incentive compatible. Observe that no player from among N_i points to a player j from among N_{i+1}, by contradiction: If it would, then player j would have to be part of the same cycle, and therefore would be in N_i. Therefore, a player in N_{i+1} cannot become part of N_i (or N_k for $k < i$) by lying, because lying only changes its outgoing arc, but not its incoming arc. Since player j gets her favorite house outside of $N_1 \cup \ldots \cup N_i$ and cannot get a house in $N_1 \cup \ldots \cup N_i$, she has no incentive to lie.

While these properties may appear strong at first sight, they are indeed trivial to achieve: Simply give her initial house to every player. This trivial mechanism can be beaten easily if I like your house better and you mine: We will simply exchange houses. This observation entails a requirement for good house allocation: A mechanism should make it impossible for a subset of players to exchange houses among themselves and be better off. Accordingly, we call an allocation a *core allocation* if no coalition (i.e., subset) of players can make all of its members better off via reallocation of their houses. Then we say that there is no *blocking coalition*. Note that a player in a coalition who does not improve, but stays in the same situation, does not need to be considered part of the coalition. Therefore, our requirement to make everybody strictly better off in a coalition is not as delicate as it might look.

4. Unique core allocation. TTCA finds the unique allocation in the core. The proof that the TTCA allocation is in the core is by contradiction. Assume there exists a blocking coalition S, a subset of all players. Let i be the first iteration in which a player j in S gets a house, i.e., the smallest i for which $N_i \cap S \neq \emptyset$. Note that player j gets her favorite house outside $N_1 \cup \ldots \cup N_{i-1}$. Since no player of S belongs to $N_1 \cup \ldots \cup N_{i-1}$, no reallocation within S can make j better off, a contradiction. To show that the allocation in the core is unique, we observe that all players in N_1 get their top choices. Hence, these players must get their top choices also in any core allocation, otherwise they would contain a blocking coalition formed by those players in N_1 who did not get their top choices. Similarly, all players in N_2 must get their houses as allocated by TTCA also in any core allocation. By induction, the same holds for all players.

2 Kidney Exchange

One might ask for which practical settings the house allocation problem arises. Kidney exchange appears to be similar to house allocation. Kidney transplantation from a living donor to a patient in need has become a routine procedure. Most often, the willingness of a donor to donate a kidney is limited to spouses, close relatives and friends. But not each kidney is suitable for each patient (blood type and other factors play a role), so the kidney of a willing donor may not be suitable for the patient. That's why kidneys should be exchanged among pairs of patient and donor, to help as many patients as possible. If for a set of pairs of patient and donor, each patient has a total preference order over all kidneys (taking into account factors such as blood type, tissue type, and many more factors), the TTCA mechanism can be used to make everybody better off.

For the reality of kidney exchange, the TTC Algorithm may not give good solutions, for two reasons. One reason is the fact that cycles produced by TTCA may be very long, but long cycles correspond to lots of surgery that should happen at the same time and at the same hospital, a difficulty or an impossibility. The condition for all involved surgical operations to happen simultaneously comes from the problem that after a willing donor's spouse got a new kidney, the formerly willing donor may no longer be willing to donate a kidney. This not only gives an unfair free kidney to his spouse, but also prevents the other donor's spouse from taking part in the kidney exchange in the future, since she has no donor to offer. Another reason is that a total preference order over all kidneys may be an overkill, and it may be preferable to simply distinguish suitable kidneys from unsuitable ones.

Therefore, a viable alternative for kidney exchange is to look for pairwise kidney exchanges in a graph of binary preferences. Instead of a directed graph, we can as well model this situation by an undirected graph, where each vertex is a patient-donor pair and each edge tells that both pairs are interested in an exchange. That is, we look for a matching of maximum cardinality in the exchange graph, where a matching is a subset of the set of edges in which no two edges share an end vertex. Now the goal is to find a maximum cardinality matching in a truthful way, where the private information of a patient-donor pair is the set of acceptable other patient-donor pairs, that is, the set of outgoing arcs in the directed graph that gives rise to the undirected graph. We aim at a matching algorithm for which reporting all outgoing arcs is a dominant strategy.

We will show that the following mechanism achieves this objective.

Maximum Matching Mechanism

1. Get a bid from each player (patient-donor pair) consisting of the acceptable edge set F_i for player i.
2. Define the set of edges on which the players agree as $E = \{(i,j) \mid (i,j) \in F_i \cap F_j\}$.
3. Return a maximum (cardinality) matching.

In order to prove truthfulness, we need to clarify which of the maximum matchings should be returned in Step 3. Maximum matchings differ in two possible ways. First, in an even length cycle, each of two alternating matchings can be chosen. No matter which one is chosen, the same patient-donor pairs get kidney exchanges, and therefore there is no room for strategizing. Second, for a single node with several incident edges as in a star, any one of these edges might be chosen, which might give rise to strategizing. One can avoid this by giving priorities to patient-donor pairs initially, according to medical criteria (e.g., how much in need is a patient, how long did she wait already) or circumstantial criteria, and by choosing the edge that matches a highest priority vertex if there is a choice. Let the priorities be reflected in the identities of the vertices. Then, Step 3 becomes:

Priority Matching Mechanism (Cont'd)

3a. Let M_0 be the set of all maximum matchings of the given graph.

3b. Loop for $i = 1, \ldots, n$:
// The order of vertices in this loop reflects their priorities
// Serve vertex i as best you can
If some matching in M_{i-1} matches vertex i, then kick out all matchings from M_{i-1} that do not match vertex i, yielding M_i.
Else leave M_{i-1} as is, yielding $M_i = M_{i-1}$.

3c. Return an arbitrary matching from M_n.

Note that since M_0 is nonempty, so is M_n. It is easy to see by induction that the priority matching mechanism is truthful in the sense that no vertex can go from unmatched to matched by reporting a proper subset of its true edge set, for every collection of true edge sets and every ordering of the vertices.

If one also takes the incentives of hospitals in to account, the situation is less satisfactory. Let us assume that there is a national (or even wider) kidney exchange office to which hospitals are required to report their patient-donor pairs, and where then matches are made (as it is the case in the United States). Each hospital has the objective to match as many of its patient-donor pairs as possible, and this objective is not well aligned with the global objective of matching as many patient-donor pairs as possible overall, as we will see in the following example. Assume two hospitals H_1 and H_2 have patient-donor pairs 1, 2, and 3 in H_1 and 4, 5, 6 and 7 in H_2. Assume that the graph has edges (1,2), (1,4), (2,5), (3,7), (5,6), and (6,7). Since for an odd number of vertices, no perfect matching exists, at least one vertex must remain unmatched. There are six matchings with three

edges each, and in some of them a vertex from H_1 is unmatched. If, however, H_1 does not report its edge (1,2) to the mechanism, there will be only one maximum matching, and this matching will match vertex 3. Therefore, H_1 has an incentive to misreport. That is, the incentive of the hospital is not in line with the incentive of society. Finding good approximate solutions is therefore an area of current research.

3 Stable Matching

The stable matching problem differs from the kidney exchange matching problem in two ways. Preferences are not binary, but each player has a total order over all alternatives, and the result of the matching is required to contain no blocking pair of vertices. Since stable matching is a workhorse for many situations and has been used for decades in settings such as assigning medical school graduates to hospitals, assigning room mates to dormitories, and many more, let us describe it independently of kidney exchange in full detail, at the classical example of matching men and women. We limit ourselves to *bipartite* stable matching, also called *stable marriage*. We are given n men and n women. Each man has a total preference order over all women, and each woman has a total preference order over all men. As an example, consider the set U consisting of three men A, B, and C, and the set V consisting of three women D, E, and F. Assume the preferences of the men are all identical lists D, E, F, where D is liked best and F least. For woman D, the preferences are A, B, C. For woman E, they are B, C, A. For woman F, they are C, A, B. The graph representing the possible matchings is a complete bipartite graph between the vertices in U and those in V.

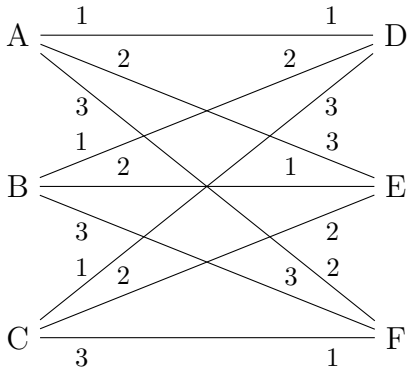

Figure 1: Visualization of the example. Edge labels denote preferences. Labels range from 1 (most preferred) to 3 (least preferred).

A maximum matching in a complete bipartite graph matches every vertex and is therefore called a *perfect matching*. Among all perfect matchings, our goal is to identify one that does not contain a blocking pair, i.e., a pair $u \in U, v \in V$ of vertices that are not matched, but prefer each other to the partners to which they are matched. Obviously, such a pair would make the matching unstable, because u could simply run away with v from their marriages and be better off.

Let us study the classical algorithm to solve this problem, the *Proposal Algorithm* by Gale and Shapley.

Proposal Algorithm

1. Initially, nobody is matched.
2. Loop until all men are matched
   Let man u propose to his favorite woman who has not rejected him yet.
   Let each woman only entertain her best offer so far and reject all others.

Observe that this algorithm is not fully determined, because it is open which man u to choose in an iteration of the loop. Still, one can see a few properties of the algorithm: 1) Over time each man goes through his preference list sequentially, from best to worst. 2) For each woman, the men that she accepts over time (and maybe later rejects) get better over time. 3) At any point in time, each man is matched to at most one woman, and vice versa. 4) The proposal algorithm terminates after at most n^2 iterations. The reason is that each man asks each woman in his list at most once. 5) The proposal algorithm terminates with a perfect matching. Otherwise, a man would have been rejected by all women. But a woman rejects a man only if she has a better man, of which there are only $n-1$. So, only $n-1$ women can end up with a better man than, and one woman would be left without a man, a contradiction. 6) The proposal algorithm terminates with a stable matching. For the sake of contradiction, consider man u not matched to woman v. There are two possible reasons for u not being matched to v: Either u never proposed to v, or u proposed to v at some point, but was rejected by v (either upon proposing or later). In the first case, u must be matched to a woman higher than v on u's list. But then the pair u, v is not blocking. In the second case, v must have rejected u for a man higher on v's list than u, so also in this case, u, v are not a blocking pair. 7) As a corollary, we now know that for every collection of preference lists a stable matching exists.

After observing these properties without having specified how the next man u to propose is chosen, let us now study how we should choose the next man to get the best result. At this point, it is not even clear what the range of possible results is that the proposal algorithm generates, and how good they are for the men and women. For man u, let $h(u)$ denote the best woman u can possibly get, i.e., the highest ranked woman on u's list that is matched to u in any stable matching. Amazingly, each man u gets $h(u)$ in the proposal algorithm, as the following theorem states:

Theorem 8.1. *For every man $u \in U$, the proposal algorithm matches u with $h(u)$.*

This in particular means that it makes no difference which man u is picked next.

Proof. For a pair u, v that is matched by the proposal algorithm, we know that any woman v' whom u prefers to v must have rejected u at some point. We only need to prove that this will always be fine i.e., that whenever a woman v' rejects a man u at some point, no stable matching pairs u and v'. This will imply the theorem. We prove the claim by induction. Initially, no woman rejected any man, so the claim holds. Now consider the rejection of a man u by a woman v'. Note that v' rejects u in favor of a better man u'. Since u' worked from the first woman of his preference list down to v', every woman that u' prefers to v' rejected him already. By the inductive hypothesis that no stable matching matches a man to a woman that rejects him in the proposal algorithm, no stable matching matches u' with a woman whom he prefers to v'. As v' prefers u' to u, and u' prefers v' to any woman he might get in any stable matching, it would be unstable to match u with v', and hence no stable matching pairs u with v'. This concludes the induction. □

Interestingly, the stable matching found by the proposal algorithm turns out to be worst for the women, a fact that we will not prove here. So, the proposal algorithm is really a *m*ale proposal algorithm that gives every man his best woman. It comes therefore as no surprise that it is truthful for the men, but not for the women, when preference lists are private:

Theorem 8.2. *The male proposal algorithm is incentive compatible for the men, but not for the women.*

Proof. No truthfulness for the women can be demonstrated by an example. Assume we are given men D, E, F and women A, B, C (for a change of names). The preferences are as follows: Man D ranks the women B, A, C. Man E ranks the women A, C, B. Man F ranks the women A, B, C. Woman A ranks the men D, F, E. Woman B ranks the men F, D, E. Woman C ranks the men D, F, E. These are the true ranks, and the male proposal algorithm matches D with B, E with C, and F with A. If, however, woman A lies and declares D, E, F instead, she gets man D, a better man for her.

Truthfulness for the men can be seen by contradiction. Assume some man u lies and improves. Let M be the stable matching that the male proposal algorithm produces for true preferences, and let M' be the matching produced for the preference lists in which u lies. Let R be the set of all those men who improve in M' as against M. Let S be the set of women matched to men in R in the matching M'. Let v be the woman that u gets in M'. Since M is stable, we know that v cannot prefer u to the man she got in M, because this would make u, v a blocking pair in M (recall that man u is better off in M' than in M, so he prefers v to the woman he gets in M). In other words, woman v prefers the man she gets in M to u. Now, if v's man in M would not improve in M', he would propose to v in M', and since v prefers him to u, v could not be matched with u in M', a contradiction. Therefore, v's man in M also improves in M', that is, belongs to R. Hence, S is not only the set of women in M' of the men in R, but also the set of women in M of the men in R. In other words, each woman in S is matched to two different men from R in matchings M and M', being better off in M than in M'. We will now show that M' cannot be stable, a contradiction that terminates the proof. We show this by looking at M and focusing on the last proposal of a man from R in the proposal process that leads to M. Call the man who proposes last $u' \in R$. This proposal must be to his woman in M, say v', and v' must accept for this proposal to be last from among men in R. We know already that $v' \in S$. Now observe that every woman in S rejects in M her man in M', because this man prefers her over his woman in M and hence went through his preference list proposing to the rejected woman earlier. Especially, v' must have rejected her man in M' already in the male proposal algorithm producing M, so v' must have had a man u'' when u' proposed to her, and she must have rejected this man for u'. Because v' only improves as time passes and she accepted u'' after rejecting her man in M', she prefers u'' to her man in M'. Note that in particular, u'' cannot be her man in M', because if it were, u'' would belong to R and would need to propose again, contradicting the assumption that the proposal by u' is the last one from among R. For this reason u'' is known to be outside R. After u'' is rejected by v', he ends up with a woman lower on his list than v'. This, however, makes M' unstable, as we will now show. Woman v' prefers u'' to her man in M', because she accepted u'' after having accepted her man in M'. Man u'' in turn prefers v' over his woman in M, because he got rejected by v' who must therefore be higher on his list. Furthermore, u'' prefers his woman in M over his woman in M', because he does not belong to R. Hence, u'' prefers v' to his

woman in M', so these two form a blocking pair, making M' not stable. □

Recommended Literature

- James Schummer and Rakesh Vohra, Algorithmic Game Theory, Chapter 10: Mechanism Design without Money, Cambridge University Press, 2007. (General introduction to the topic)
- Tim Roughgarden, Lecture Notes for 364A: Algorithmic Game Theory, Lecture #9: Beyond Quasi-Linearity, 2015. (House allocation)
- Tim Roughgarden, Lecture Notes for 364A: Algorithmic Game Theory, Lecture #10: Kidney Exchange and Stable Matching, 2015. (Kidney exchange and stable matching)
- Lloyd S. Shapley and Herbert Scarf, On Cores and Indivisibility. Journal of Mathematical Economics, 1(1):23–28, 1974. (TTC Algorithm)
- Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Kidney Exchange. Quarterly Journal of Economics, 119(2):457–488, 2004. (Original results on kidney exchange)
- David Gale and Llyod S. Shapley, College Admissions and the Stability of Marriage, American Mathematical Monthly, 69: 9–14, 1962. (Man Proposal Algorithm)