601.436/636 Algorithmic Game Theory **Lecturer:** Michael Dinitz
Topic: Mechanism Design Without Money **Date:** 4/19/22

23.1 Introduction

So far, all of our mechanisms have involved money: we incentivize players to tell us the truth by using payments to align their incentives. But sometimes it is immoral or illegal to use money. Often this makes our task impossible: without money there just isn't any way to incentivize truth telling. But sometimes we can still design good mechanisms. Classical examples of this are things like organ transplants/donations (particularly kidney donations) and matching problems (most famously, matching new residents and hospitals). Today we'll talk about a few of these, but will only really be able to scratch the surface.

23.2 House Allocation

This is maybe not the most realistic problem ever, but is one of the most classical settings. There are n agents, each of which owns a house. Each agent has a total ordering over the n houses, not necessarily preferring their own. Can we reallocate the houses to make everyone better off, and in a way that is incentive compatible? Slightly more formally, a "bid" is an ordering of the houses, and the agents may lie and bid an ordering other than their true one. We want a way of reallocating the houses where everyone is incentivized to tell the truth.

Note that there's nothing special here about houses – this applies more generally to reallocating any set of items.

23.2.1 Top Trading Cycle Algorithm

The following algorithm is known as the *Top Trading Cycle Algorithm* (TTCA).

- Let $A = [n]$ be the agents.
- While $A \neq \emptyset$:
  - For each $i \in A$, let $f(i)$ be the favorite remaining house (favorite house belonging to an agent in A). Let $E = \{(i, f(i)\}_{i \in A}$. This gives us a natural directed graph $G = (A, E)$ in which every node has outdegree 1. This outdegree property implies that G contains at least one directed cycle (loops count), and each node is in at most one directed cycle.
  - For each directed cycle C, reallocate in the obvious way: each agent in C gives their house to their predecessor in C (and so gets the house of their successor, which was their favorite remaining house).
  - Remove all agents who were reallocated from A.

Note that this gives a feasible allocation (every agent has exactly one house when the algorithm terminates), and if an agent bids truthfully then they're never worse off because they participated in the algorithm (since whenever they get allocated they get their successor, who they prefer at least as much as themselves).

23.2.2 Incentive Compatibility

Let's start by proving incentive compatibility.

Theorem 23.2.1. *Top Trading Cycle is incentive compatible.*

Proof. Consider some agent j and fix bids from all other agents. We need to show that the best thing for j to do is bid its true ordering.

Suppose that we run the algorithm with j bidding its true ordering. Let N_k denote the agents (re)allocated in iteration k of the algorithm, and let G_k denote the graph in the k'th iteration. Suppose that j is in N_i. This means that if j tells the truth, it gets its favorite house among all houses not in $N_1 \cup N_2 \cup \ldots N_{i-1}$.

We claim that if some agent ℓ has a path P to j in G_k, then this path is also in $G_{k'}$ for all $k \leq k' \leq i$. This is easy to show by induction. For the base case, it is true when $k' = k$. For the inductive step, let $k \leq k' < i$. The induction hypothesis implies that the path is still there in $G_{k'}$ and no vertex in the path can be part of a cycle in $G_{k'}$ (or else j would be in that cycle). Thus the path is still there in $G_{k'+1}$. Thus any agent who has a path to j at some point is allocated no earlier in j, i.e., is not in $N_1 \cup \cdots \cup N_{i-1}$.

If j lies, then it only makes a difference if this creates a cycle in some G_k with $k \leq i$. But by the claim, if it creates such a cycle then all vertices of this cycle are not in $N_1 \cup \cdots \cup N_{i-1}$. Thus j cannot do better by lying. □

23.2.3 Quality

Now that we know TTCA is incentive compatible, the obvious question is whether this reallocation is any good. It's not entirely obvious how to quantify or define this, since we do not have utilities. We'll do this through a new definition, which is essentially just a strong Nash but without utilities.

Definition 23.2.2. *A* blocking coalition *of an allocation is a subset of agents which reallocate among themselves so that no one is worse off and at least one agent is better off.*

Definition 23.2.3. *A* core allocation *is an allocation with no blocking coalitions.*

Note that since we have total orderings, if an agent gets reallocated it is either worse off or better off – there are no ties. Since we proved incentive compatibility, we only need to analyze what happens when all bids are truthful. Let N_i be the agents that are allocated in iteration i of the algorithm when all players bid truthfully.

Theorem 23.2.4. *Top Trading Cycle returns a core allocation.*

Proof. Consider some coalition $S \subseteq [n]$ of agents and reallocation among them. This reallocation consists of a set of disjoint directed cycles (create an edge from u to v in S if u gives their house to

v). If in every cycle all agents are in the same N_i, then since they already had their favorite house not in $N_1 \cup \cdots \cup N_{i-1}$ the reallocation has made them worse off. So consider a cycle with agents in different iterations. Since it's a cycle, there must be some edge from a node $u \in N_i$ to $v \in N_j$ where $j < i$. Then this means that v is getting a house from some N_i with $i > j$, but it was already getting its favorite house not in $N_1 \cup \cdots \cup N_{j-1}$. So this reallocation makes v worse off. Thus S is not a blocking coalition. □

Theorem 23.2.5. *The allocation returned by Top Trading Cycle is the only core allocation.*

Proof. Let's prove this by induction. In the allocation from TTC, all agents in N_1 get their favorite house. So in any core allocation, all agents in N_1 must also get their favorite house or else N_1 is a blocking coalition. Thus any core allocation must agree with TTC on agents in N_1.

Now suppose that every core allocation agrees with TTC on the agents in $N_1 \cup \cdots \cup N_{i-1}$. Then every agent in N_i gets their preferred house outside of $N_1 \cup \cdots \cup N_{i-1}$ in TTC, and so much also get the same house in any core allocation or else N_i would be a blocking coalition. □

So there is a sense in which the Top Trading Cycle algorithm is optimal: it always returns the only core allocation.

23.3 Stable Matching

Let's talk about a similar but different setting, known as *stable matching*. This turns out to be a super important setting, most famously in matching residents and hospitals in the US where a slight modification of this is used.

We are given two sets of agents U and V, each of size n. Think of U as residents and V as hospitals (in practice there are more residents than hospitals, which requires slightly changing the setup, but it's not too important so we'll think of U and V has having the same size). Every agent in U has a total ordering (preference) over V, and every agent in V has a total ordering over U.

Definition 23.3.1. *A* stable matching *is a perfect bipartite matching between U and V (i.e., a bijection $\pi : U \to V$) such that if $u \in U$ and $v \in V$ are not matched then either u prefers $\pi(u)$ to v or v prefers $\pi^{-1}(v)$ to u.*

In other words, in a stable matching there is no pair that prefers to ignore the matching and pair up themselves.

Stable matchings are clearly nice objects, so there are a few obvious questions. Does a stable matching always exist? If so, can we compute it? If so, is there an incentive compatible mechanism which computes it? As we'll see, the answers are yes, yes, and sort of.

23.3.1 Deferred Acceptance Algorithm

- While there is an unmatched $u \in U$:
    - u sends a proposal to favorite $v \in V$ that has not yet rejected u.

  - If v is unmatched, tentatively match u and v
  - Else v is tentatively matched to some $u' \in U$. Then v chooses to tentatively match to whichever of u, u' it prefers, and rejects the other one.

- Make all tentative matches final.

23.3.2 Analysis

Let's show that this algorithm computes a stable matching. We'll do this in three steps: showing that it terminates, that it computes a perfect matching (bijection), and that this matching is stable.

Lemma 23.3.2. *There are at most n^2 iterations of the algorithm.*

Proof. In every iteration some agent in U proposes to an agent in V. But no agent in U ever proposes to the same agent in V twice. So there can be at most n^2 total proposals, and thus at most n^2 iterations. □

Lemma 23.3.3. *The algorithm terminates with a perfect matching.*

Proof. The algorithm clearly maintains a matching throughout, so we just need to prove that it is perfect. Suppose it is not. Then there is some $u \in U$ not matched, so u must have been rejected by all $v \in V$. This implies that all $v \in V$ get at least one proposal. But once a $v \in V$ gets at least one proposal, it is matched and stays matched throughout the rest of the algorithm (possibly to different nodes in U). Thus all $v \in V$ are matched when the algorithm terminates, which implies that all $u \in U$ are also matched and the matching is perfect. □

Lemma 23.3.4. *The returned perfect matching is stable.*

Proof. Consider some $u \in U$ and $v \in V$ who are not matched. If u never proposed to v then u is matched to someone that it prefers to v, and so u would not want to switch to v. If u did propose to v, then it was rejected (either then or later) for someone that v preferred more, and so v would not want to switch to u. □

It turns out that there can be many stable matchings, and it is again a bit unclear how to compare them. It's also unclear whether this algorithm is "incentive compatible" in some sense. It turns out that this algorithm is actually great for agents in U, in the following sense. For each $u \in U$, let $h(u)$ be the most preferred agent in V that u is matched to in *any* stable matching. Then our deferred proposal algorithm actually is precisely that matching.

Theorem 23.3.5. *The deferred proposal algorithm matches each $u \in U$ to $h(u)$.*

Proof. Let R be the set of all pairs (u, v) such that v rejects u at some point in the algorithm. So if u is matched to v' at the end, then $(u, v) \in R$ for all v that u prefers to v'. So we just need to show that if $(u, v) \in R$ then there is no stable matching in which u is matched to v.

Let's prove this by induction over the iterations, i.e., show that it is an invariant of the algorithm. Initially no proposal has been rejected so $R = \varnothing$, and thus it is true that if $(u, v) \in R$ then there is no stable matching in which u and v are matched. For the inductive step, consider an iteration in which u was rejected by v in favor of some other u' (and thus either u or u' proposed to v). So after this iteration u' is matched to v, and thus before the iteration $(u', v') \in R$ for all v' that u' prefers to v. Thus by induction there is no stable matching in which u' is paired to an agent they prefer to v. Combining all of this, we get that v prefers u' to u and u' prefers v to anyone they could be matched to in any stable matching. Thus if u were matched to v in some stable matching, u' and v would deviate to each other, contradicting the definition of stable matching. □