CS 286r: Matching and Market Design **Lecture 2**
Combinatorial Markets, Walrasian Equilibrium, Tâtonnement

Matching and Money

Recall: Last time we described the Hungarian Method for computing a maximum-weight bipartite matching. In principle, we could use this algorithm to determine outcomes for a market. For example:

- Housing markets
- Assignment of workers to jobs

Note: The algorithm will find an efficient matching, but it might be that not every participant is happy with the outcome. If we were to resolve real economic markets using the Hungarian method (or any other algorithm), the participants might benefit by lying about their preferences.

Goal: Instead of simply choosing a market outcome, design a market where the participants are individually incentivized to select the optimal matching.

Idea: Use payments!

Combinatorial Markets

In a combinatorial market, there is a set N of n *agents* (buyers) and a set M of m *goods* (or *items*). There is one indivisible copy of each good. Think: houses.

Def: A *valuation function* assigns a non-negative value to each set of goods: $v : 2^M \to \mathbb{R}_{\geq 0}$. Each agent i has a valuation function v_i. We will assume valuation functions are

- *monotone*: $v(S) \leq v(T)\ \forall S \subseteq T$, and
- *normalized* so that $v(\emptyset) = 0$.

Def: An *allocation* x is a partition of the goods among the agents, with possibly some goods left unallocated. We write x_i for the set of goods allocated to agent i.

Def: The *social welfare* of an allocation x is $\sum_i v_i(x_i)$. An allocation that maximizes social welfare is said to be *efficient*.

Goal: Find an allocation x that maximizes social welfare. We will call this the *allocation problem*.

Special Case: Matchings

Note: In the special case that each agent can be allocated at most one item, an allocation is precisely a matching. In this case, a valuation function simply assigns a value to each item. The allocation problem is equivalent to finding a maximum-weight bipartite matching.

To see that this is a special case of a combinatorial market, suppose that each valuation function is *unit-demand*. A valuation v is unit-demand if it assigns a value $v(j)$ to each item $j \in M$, and then for any set of items S we have

$$v(S) = \max_{j \in S} v(j).$$

That is, each agent gets value from at most one of the items allocated to him. When valuation functions are unit-demand, it is without loss of generality to allocate at most one item to each agent.

Example:

We will use this running example throughout. There are 3 items, $\{a, b, c\}$, and 3 agents:

- Alice: $v(a) = \$2$, $v(b) = \$3$, $v(c) = \$0$
- Bob: $v(a) = \$0$, $v(b) = \$2$, $v(c) = \$4$
- Charlie: $v(a) = \$0$, $v(b) = \$4$, $v(c) = \$5$

Alice and Bob are both unit-demand agents. (Alice $\leftarrow \{a\}$, Bob $\leftarrow \{c\}$, Charlie $\leftarrow \{b\}$) is an allocation, with social welfare 10.

Walrasian Equilibrium

Left to their own devices, each agent would naturally want to take the item they value the most, but this might cause some items to be overdemanded. To coordinate the agents' preferences, we introduce prices.

Imagine that every good $j \in M$ has a *price* $p_j \geq 0$. If agent i is allocated a set of goods x_i, he must pay $\sum_{j \in x_i} p_j$ in exchange for receiving those goods.

Def: the *utility* of agent i, given that he is allocated set x_i, is $v_i(x_i) - \sum_{j \in x_i} p_j$.

Each agent wants to maximize his or her own utility.

Def: the *demand correspondence* of agent i at prices p, $D_i(p)$, is the set of utility-maximizing sets of goods. That is, $D_i(p)$ equals

$$\{S \,:\, v_i(S) - \sum_{j \in S} p_j \geq v_i(T) - \sum_{j \in T} p_j \; \forall T \subseteq M\}.$$

Def: a *Walrasian Equilibrium* is a choice of item prices p, plus an assignment x, such that

- every agent is allocated a demanded set: $x_i \in D_i(p)$ for all i.
- every item with positive price is sold: if $j \notin x_i$ for all i, then $p_j = 0$.

Example:

Consider our running example. Price vector $(p_a = 0, p_b = 1, p_b = 2)$, along with allocation (Alice $\leftarrow \{a\}$, Bob $\leftarrow \{c\}$, Charlie $\leftarrow \{b\}$), together form a Walrasian Equilibrium.

Theorem 0.1 *If (x, p) is a Walrasian equilibrium, then x maximizes social welfare.*

Proof: Say (x, p) is a WE, and let y be any other allocation. For each agent i, we know $v_i(x_i) - \sum_{j \in x_i} p_j \geq v_i(y_i) - \sum_{j \in y_i} p_j$. Take a sum over all agents to get

$$\sum_i v_i(x_i) - \sum_i \sum_{j \in x_i} p_j \geq \sum_i v_i(y_i) - \sum_i \sum_{j \in y_i} p_j.$$

Since all items with positive price are allocated under x, we actually have that $\sum_i \sum_{j \in x_i} p_j$ is equal to $\sum_{j \in M} p_j$, which cannot be less than $\sum_i \sum_{j \in y_i} p_j$. So

$$\sum_i \sum_{j \in x_i} p_{x_i} \geq \sum_i \sum_{j \in y_i} p_{y_i},$$

from which we conclude

$$\sum_i v_i(x_i) \geq \sum_i v_i(y_i),$$

so x must be optimal. □

Tâtonnement

The following simple procedure is useful for constructing a Walrasian Equilibrium. It is called "Tâtonnement," which translates roughly to "grasping around blindly." The basic idea is to raise prices on over-demanded items until the market reaches equilibrium.

We will begin by presenting a simple version that modifies prices in discrete jumps, but obtains only an approximate equilibrium. Later we will see how to turn this into a method for finding exact Walrasian equilibria.

Def: the ϵ-approximate demand correspondence of agent i given prices p, $D_i^\epsilon(p)$, is the set of sets of goods that maximize agent i's utility, up to an additive ϵ error. That is, $D_i^\epsilon(p)$ is equal to

$$\{S : v_i(S) - \sum_{j \in S} p_j \geq v_i(T) - \sum_{j \in T} p_j - \epsilon \; \forall T \subseteq M\}$$

Def: an ϵ-approximate Walrasian equilibrium is an allocation x and price vector p such that $x_i \in D_i^\epsilon(p)$ for all i, and all unsold items have price 0.

Algorithm:

1. Let $\delta > 0$ be arbitrarily small.
2. Start with $p_j = 0$ for all $j \in M$, and $x_i = \emptyset$ for all $i \in N$.
3. If $x_i \in D_i^{\delta \cdot m}(p)$ for all i, stop and return (x, p).
4. Suppose $x_i \notin D_i^{\delta \cdot m}(p)$. Pick $y_i \in D_i(p)$.
5. For each item $j \in y_i$, increase p_j by δ.
6. Allocate y_i to agent i: set $x_i = y_i$, and set $x_k = x_k \backslash y_i$ for all $k \neq i$.
7. Continue on line 3.

Tâtonnement does not always succeed in finding an equilibrium; we will see an example of that later. However, for the special case of matching markets, it turns out that Tâtonnement does always converge to a Walrasian equilibrium.

Theorem 0.2 *The Tâtonnement process must terminate. In a matching market, it terminates at a $(\delta \cdot m)$-approximate Walrasian equilibrium.*

Proof: Prices only rise, and some price rises on each iteration of the algorithm. The algorithm must therefore terminate eventually, since an item whose price is sufficiently high is not included in any agent's demanded set(s).

By line 3, if the algorithm terminates, it must be that $x_i \in D_i^{\delta \cdot m}$ for each agent i. So it only remains to show that each unallocated item has price 0. We claim that once an item is allocated, it never again becomes unallocated. This will complete the proof, since then the only unallocated items are items that were never allocated, and thus never had their prices incremented on line 5.

Suppose for contradiction that the algorithm unallocates some item j on some iteration of lines 3-7; say iteration t. This can only occur if an agent i with $j \in x_i$ is selected in line 4, but $j \notin y_i$. In this case, item j would be unallocated on line 6. However, in a matching market, allocations are singletons; it must therefore be that $x_i = \{j\}$. Moreover, it must be that $\{j\}$ was the allocation assigned to agent i on a previous iteration, say $t' < t$. So, on iteration t', it must have been that $\{j\} \in D_i(p)$. Since the prices of other items can only increase, and p_j increased only by δ on iteration t', $\{j\}$ must be in $D_i^\delta(p)$ on iteration t. This contradicts the fact that agent i was selected on line 4 in iteration t. $\square$

Bonus Material: Exact Walrasian Equilibrium

The Tâtonnement algorithm described in the previous section only finds an approximate Walrasian equilibrium. We can make this approximation arbitrarily good by taking δ as small as desired. In the limit as $\delta \to 0$, this has the net effect of allowing prices to rise continuously. In this limit, the algorithm finds an exact Walrasian equilibrium for matching markets.

To formalize this, suppose (x^δ, p^δ) is the output of the Tâtonnement process for a given choice of δ, say restricted to $\delta \in (0, 1]$. Consider the limit $\delta \to 0$. As there are only finitely many choices of allocations, some allocation x must occur for infinitely many δ. Restrict attention to only those values of δ for which $x^\delta = x$; this produces a subsequence of values of δ with limit 0. For each of these infinitely many δ's there is a price vector p^δ; moreover, these prices vectors lie in a bounded range (since no price can be greater than all agents' values for all items). There must therefore be a subsequence of δ's for which the associated price vectors converge to some vector of limit prices, say p, as $\delta \to 0$. But this then implies that (x, p) is an ϵ-approximate Walrasian equilibrium for all sufficiently small ϵ (since, as $\delta \to 0$, it can be made arbitrarily close to a $(m \cdot \delta)$-approximate equilibrium). Therefore (x, p) is an exact Walrasian equilibrium as well.

Faster Tâtonnement

The Tâtonnement method involves continuously raising prices, so it may take a long time to run. We can improve the runtime for matching markets by being more careful about the order in which we raise prices, and by grouping together many price-increment operations.

We will think of a matching market as being represented by a weighted bipartite graph, with agents on one side and goods on the other. The weight of an edge between agent i and good j will be $v_i(j)$, the value of agent i for good j.

Def: Given a weighted bipartite graph between agents and goods, plus a price vector p over the goods, the utility-maximizing graph, $E(p)$, consists of all edges (i, j) for which $\{j\} \in D_i(p)$. That is, $E(p)$ contains all edges between agents and utility-maximizing goods for those agents.

An observation about the Tâtonnement process is that each agent is trying to obtain a utility-maximizing good. If an agent's assigned good is taken away on some iteration of the algorithm, then that agent will switch to a different utility-maximizing good if there is one; otherwise he will reaquire the previous good and increase its price. This process will continue until a new good becomes utility-maximizing for some agent, which must occur eventually if prices keep increasing. We can therefore think of the Tâtonnement process as attempting to match every agent to a utility-maximizing good, and if this is not possible then the prices of the currently desired items are raised until new items become utility-maximizing (that is, new edges are added to $E(p)$). With this interpretation in mind, we can speed up the Tâtonnement process by "looking ahead" to see how much we would need to increase prices in order for new edges to be added to the utility-maximizing graph. This leads to the following alternative implementation of Tâtonnement, which runs in polynomial time.

Algorithm:

1. Start with all prices set to 0.
2. Let M be a maximal matching in $E(p)$. If M is perfect, stop and return matching M and prices p.
3. Otherwise, find an agent i not matched in M.
4. "Direct" $E(p)$: all edges in M point toward the agents, and all other edges point toward the items.
5. Raise the prices of all items reachable from i in this directed graph, until a new edge is added to $E(p)$.
6. Go to line 2.

Example: Consider our running example.

- When all prices are set to 0, $E(p)$ contains edges $(A, b), (B, c)$, and (C, c). Let $M = \{(A, b), (C, c)\}$. Agent B is not matched. Item c is reachable from B, so we will raise price p_c.
- Once p_c is raised to 1, edge (C, b) becomes part of $E(p)$. The maximal matching in $E(p)$ becomes $\{(B, c), (C, b)\}$. Agent A is not matched. Items b and c are both reachable from A, so we will raise p_b and p_c in tandem.
- When the price vector becomes $(0, 1, 2)$ (i.e., p_b and p_c are raised by 1), edge (A, a) becomes utility-maximizing. So $E(p)$ becomes $\{(A, a), (A, b), (B, c), (C, b), (C, c)\}$. The maximal matching in $E(p)$ becomes $\{(A, a), (B, c), (C, b)\}$. This is a perfect matching, so the algorithm terminates.
- The final output of the algorithm is allocation (Alice $\leftarrow a$, Bob $\leftarrow c$, Charlie $\leftarrow b$) at prices $(0, 1, 2)$.

Claim: This procedure terminates in at most nm iterations, and finds a Walrasian equilibrium.

Proof: (sketch) Each iteration causes at least one edge to be added to $E(p)$, and once an edge is added to $E(p)$ it never stops being in $E(p)$. So there can be at most nm iterations. That the algorithm finds a Walrasian equilibrium follows from equivalence with Tâtonnement. □

Observation: This algorithm is precisely the Hungarian method! The prices correspond to the dual variables on the "goods" side, and the agent utilities correspond to the dual variables on the "agent" side.

So, in fact, the Hungarian method implements the optimal weighted matching as a market outcome, when the dual variables are used as item prices.

Beyond Matching

The Tâtonnement algorithm is defined for arbitrary combinatorial markets, not just matching markets. However, in general, a WE is not always guaranteed to exist. Why does Tâtonnement fail?

Recall our proof that Tâtonnement finds an approximation Walrasian equilibrium for matching markets: we needed to show that once an item is allocated, it is never unallocated. This is true for matching markets, but is not true in general.

Example: Suppose our market has two goods $\{L, R\}$, a left shoe and a right shoe.

- Alice would like to purchase the shoes only as a pair. Her valuation is $v_A(\{L, R\}) = 5$, but $v_A(L) = v_A(R) = 0$.

- Bob is interested in any single shoe (perhaps Bob is a dog). His valuation is $v_B(L) = v_B(R) = v_B(\{L, R\}) = 3$.

In this example, Tâtonnement does not find a Walrasian equilibrium. Consider the simple Tâtonnement process, with a small price increment δ. Starting at prices 0, Alice will choose both shoes on every iteration, and Bob will choose whichever shoe is cheaper. This will continue until the price of each shoe rises above \$2.50. At this point, Bob will take one shoe from Alice (say the left), raising its price further, and then Alice will discard her remaining shoe and take her demanded set which is $\emptyset$. The resulting allocation is (Alice $\leftarrow \emptyset$, Bob $\leftarrow \{L\}$) at prices (\$2.50+, \$2.50+). This is not an approximate Walrasian equilibrium, since the right shoe is unallocated but has a positive price.

Proving a Walrasian Equilibrium does not Exist

Question: How do we prove that a combinatorial market does not have a WE?

One approach uses the theory of linear relaxations. The assignment problem can be expressed as the following mathematical program. The variables are $x_{i,S}$ for $i \in N$ and $S \subseteq M$, which we interpret as indicator variables: $x_{i,S} = 1$ means that agent i is allocated set S, and $x_{i,S} = 0$ means that S is not the set allocated to agent i.

Allocation Program:

$$\begin{aligned} \max \quad & \sum_{i,S} v_i(S) \cdot x_{i,S} \\ \text{s.t.} \quad & \sum_{S \subseteq M} x_{i,S} \leq 1 \text{ for every } i \in N \\ & \sum_{i, S \ni j} x_{i,S} \leq 1 \text{ for every } j \in M \\ & x_{i,S} \in \{0, 1\} \text{ for every } i \in N, S \subseteq M \end{aligned}$$

The program describes the space of all valid assignments. The first constraint guarantees that every agent is only allocated one set; the second constraint guarantees that every item is only allocated once.

Consider relaxing the program to allow fractional assignments. This corresponds to a linear program, known as the configuration LP. The difference between this program and the previous one is that the variables $x_{i,S}$ are allowed to take on any fractional value in $[0, 1]$.

Configuration LP:

$$\begin{aligned} \max \quad & \sum_{i,S} v_i(S) \cdot x_{i,S} \\ \text{s.t.} \quad & \sum_{S \subseteq M} x_{i,S} \leq 1 \text{ for every } i \in N \\ & \sum_{i, S \ni j} x_{i,S} \leq 1 \text{ for every } j \in M \\ & x_{i,S} \in [0, 1] \text{ for every } i \in N, S \subseteq M \end{aligned}$$

Since the linear relaxation allows more possible solutions, its maximum value can only be larger than the optimal welfare of the (integral) allocation problem. We already saw that a Walrasian equilibrium (x, p) achieves the optimal social welfare among all integral allocations, but how does its social welfare compare to the (possibly larger) optimal social welfare among all fractional allocations? As it turns out, a Walrasian equilibrium is also optimal for the LP relaxation.

Claim: A Walrasian Equilibrium maximizes social welfare among all fractional assignments.

Proof: The proof is very similar to the proof of optimality over integral assignments. See the assigned reading.

This claim is somewhat surprising, since the assignment of a Walrasian equilibrium is not fractional! So a corollary is that a Walrasian equilibrium can exist only if the optimal fractional assignment is, in fact, integral.

Corollary 0.3 *A combinatorial market has a Walrasian Equilibrium only if the integral optimal social welfare and the fractional optimal social welfare are the same.*

Note: In fact, it turns out that this is an if and only if condition. We will not cover the proof (which uses the theory of duality), but you can find it in the assigned reading.

Example: Consider the two-shoes example from before. We will prove that it does not have a Walrasian equilibrium.

The welfare-optimal assignment is to give both shoes to Alice; this achieves social welfare 5.

Consider the following fractional assignement: $x_{A,\{L,R\}} = \frac{1}{2}$, $x_{B,\{L\}} = \frac{1}{2}$, $x_{B,\{R\}} = \frac{1}{2}$. This is a valid fractional assignment: the total weight of all allocations that include L is at most 1, and similarly for R; and the total weight of all allocations to Alice is at most 1, and similarly for Bob.

The value of the fractional assignment is $\frac{1}{2}v_A(\{L,R\}) + \frac{1}{2}v_B(L) + \frac{1}{2}v_B(R) = \frac{11}{2} > 5$. Since the optimal fractional assignment generates more value than the optimal integral assignment, a Walrasian equilibrium does not exist.

Example: Modify the two-shoes example so that Alice has value 7 for the pair of shoes, and Bob still has value 3 for any one shoe. In this case, the allocation (Alice $\leftarrow \{L,R\}$, Bob $\leftarrow \emptyset$) with prices ($p_L = 3, p_R = 3$) is a Walrasian equilibrium. This allocation generates welfare 7. This is the optimal welfare even among all fractional assignments.