# Mechanism Design without Money

Shikha Singh

## Contents

# 1 Serial Dictatorship

Consider a one-sided matching markets with $N = \{1, \ldots, n\}$ participants and $M = \{1, \ldots, m\}$ items such that each participant $i \in N$ has a complete and strict preference order $P_i$ over $M$.

Our goal is to find a matching between participants and items.

**Definition 1.** A matching $\mu \subseteq N \times M$ is a set of ordered pairs $(i, j)$ where participant $i$ is a participant *matched* to item $j$ such that the following conditions hold:

- No participant $i$ is matched to more than one item $j$ in $M$
- No item $j$ is matched to more than one participant $i$ in $M$

If you think of the input as a complete bipartite graph with vertex sets $N$ and $M$, then a matching $M$ is a subset of edges such that no two edges in $M$ share a vertex.

The first algorithm we consider is the serial dictatorship algorithm.

**Serial Dictatorship Algorithm.** The serial dictatorship algorithm orders the participant's turn using some natural order (based on seniority, or randomly) and lets each participant take their favorite item among the ones remaining. More specifically, it works as follows.

- Using a one-to-one and onto ranking function $r : N \to \{1, \ldots, n\}$ that assigns a rank $r(i)$ to each participant and let $M = \{1, \ldots, m\}$.
- For $j \in \{1, \ldots, n\}$, do
  - Consider the preference list $P_i$ of participant $i = r(j)$
  - Give $i$ their top-ranked item $k$ in $P_i$ from the remaining items $M$, that is, add $(i, k)$ to matching $\mu$.[1]
  - Remove $k$ from $M$

We prove the following two properties of the serial dictatorship algorithm.

**Lemma 1.** The matching $\mu$ output by the serial dictatorship algorithm is the unique Pareto optimal matching.

[1] If $m < n$ and no item is left, let $k = \emptyset$.

**Proof.** Let $\mu' \subseteq N \times M$ be a matching that is different than $\mu$. Let $\mu(i)$ and $\mu'(i)$ denote the match of a participant $i \in N$ in the two matchings. Let $\mu(i) > \mu'(i)$ denote the condition that $i$ strictly prefers their matching in $\mu$ over $\mu'$.

If there exists a participant $i$ such that $\mu'(i) < \mu(i)$, then $\mu'$ cannot Pareto dominate $\mu$. Now suppose that for each participant $i \in N$, $\mu'(i) \geq \mu(i)$.

We prove that in this case both matchings are exactly the same, that is, $\mu' = \mu$. Both matchings are the same at the beginning before any participant is matched. Suppose they are identical up until it is the turn of agent $i$ with rank $t$ to choose. The number of items $M$ remaining is the same for both matchings, because the same items have been matched up until now. In matching $\mu$, $i$ is matched to their most preferred remaining item from $M$. If $\mu'(i)$ is different $\mu(i)$, then $\mu'(i) < \mu(i)$ which is a contradiction. Thus, $\mu'(i) = \mu(i)$.

As any matching does not Pareto dominate $\mu$ is identical to $\mu$, we have proved that $\mu$ is the unique Pareto optimal matching. □

**Lemma 2.** The serial dictatorship is **dominant strategyproof**—that is, truthful reporting of preferences is a dominant strategy for all participants.

**Proof.** Consider an arbitrary participant $i$ whose rank is $t$ in the algorithm and fix $P_{-i}$. Let $M_t$ be the number of items remaining at time step $t$ when $i$ is about to choose. Let $P_i$ be the truthful preference of $i$. Let $j$ be the rank of $i$'s most preferred item that is in $M_t$. Thus, any item in $P_i$ before rank $j$ is not attainable for $i$ (regardless of the preference $i$ chooses to report) since $P_{-i}$ and the rank function stays fixed. Since $i$ gets their favorite item among those at rank $j$ and above, truthful reporting is a weakly dominant strategy for $i$. □

**Takeaways and Conclusion.** Serial dictatorship is Pareto-optimal and strategyproof. However, it can be unfair if the rank function used is not based on a natural ordering. Random Serial Dictatorship (RSD) is an algorithm that chooses the rank ordering uniformly at random and used in situations where no natural ordering exists.

Note that the algorithm is no longer strategyproof if participants are allowed to only report partial preference lists, e.g. their top $d < m$ preferences. Consider the example of course registration where students can only preregister for 4 courses. In such cases, participants are incentivized to strategize by guessing the lottery order and others' preferences.

## 2 Top-Trading Cycle

The top-trading cycle algorithm can be used in both one-sided and two-sided matching markets. We first describe the top-trading cycle algorithm for the housing exchange problem.

Consider the a housing allocation problem with $n$ agents. Each agent $i$ has a house $h_i$ and a complete preference ranking $P_i$ over the houses of all $n$ agents (including their own house). The goal is to re-allocate houses within the $n$ agents so that each agent is matched to a house they prefer at least as much as their own house and potentially they strictly prefer over their own house. The top-trading cycle for the problem is described below.

- Each agent simultaneously points to the agent who has their favorite house (among those that are remaining). This includes a directed graph $G$ where each node has out-degree at least 1. Note that self-loops are allowed as agents can point to themselves. Thus, $G$ has at least one cycle $C$.
- For each agent in cycle $C$, give them the house they are pointing to. Remove all such agents and their houses. We repeat the above process until no agents remain.

We prove the following invariant about TTC that is helpful in establishing its properties.

**Invariant 1.** (TTC Invariant) Let $N_k$ be the set of agents removed in the $k$th iteration of the TTC algorithm. Every agent of $N_k$ receives their favorite house outside of the houses owned by $N_1 \cup N_2 \cup \cdots \cup N_{k-1}$, and the original owner of any house allocated in this round is also in $N_k$.

**Proof.** Fix an agent $i$ and preferences reported by others. Define the sets $N_k$ as in the TTC invariant. Suppose $i \in N_j$ when $i$ is truthful. Regardless of $i$'s preferences, $i$ cannot get a house originally owned by $N_1 \cup \cdots \cup N_{j-1}$. Suppose $i$ wants a house owned by $\ell \in N_k$, where $k \in \{1, 2, \ldots, j-1\}$. To get this house, $\ell$ must point to $i$ in iteration $k$ or earlier, but this is not the case if $i \notin N_k$. Truthful reporting gets $i$ the best possible house they can achieve and thus is dominant strategyproof. □

Top-trading cycle is not only Pareto-efficient, it has a stronger property that it produces the unique *core* allocation.

**Definition 2.** A subset $S \subseteq \{1, \ldots, n\}$ is a **blocking coalition** if members of $S$ can trade houses amongst themselves such that at least one member is better off without making any

member of $S$ worse off. If $S = N$, this property is the same as Pareto optimality. An allocation is core if there is no such blocking coalition.

**Theorem 1.** For any house allocation instance, the output computed by the TTC algorithm is the unique core allocation.

**Proof.** We prove this in two parts.

**Claim 1.** Any core allocation must agree with TTC.

Let $N_j$ be defined by the TTC invariant. All agents of $N_1$ receive their first choice: this must be true in any core allocation. If not, the agents of $N_1$ can internally reallocate and can make everyone strictly better off. Similarly, all agents of $N_2$ receive their top choice outside $N_1$. Given that every core allocation agrees with TTC for agents in $N_1$, such an allocation must also agree for agents in $N_2$. We can continue inductively for all $N_j$. Thus, any core allocation must agree with TTC.

**Claim 2.** TTC allocation is a core allocation.

Consider an arbitrary subset $S$ of agents. Let $\ell = \min\{j \mid S \cap N_j \neq \emptyset\}$ be the earliest round in which a member of $S$ receives their house under TTC. Consider any agent $i \in N_\ell \cap S$. Then $i$ receives their favorite house among those not obtained by $N_1, \ldots, N_{\ell-1}$. No member of $S$ received a house in rounds 1 through $\ell - 1$, that is,

$$N_j \cap S = \emptyset \quad \text{for } j = 1, \ldots, \ell - 1$$

Because round $\ell$ is the first time anyone in $S$ is allocated a house, no reallocation within $S$ can make anyone strictly better off. □

## 2.1 Case Study: School Choice

Abdulkadiroğlu and Sönmez paper [?] studies the application of School Choice as a matching market. They generalize the TTC to two-sided markets as follows and apply to assign students to schools as follows.

*In each step, each school with remaining capacity points to the unmatched student with most priority, and each unmatched student points to the most preferred school with remaining capacity. Paths alternate between students and schools, and "trading on a cycle" corresponds to each student on the cycle being matched with its requested school.*

They argue that it is better than using deferred acceptance as the output of TTC is always Pareto efficient, while DA may not be. In particular, they prove the following properties of TTC for school choice.

1. **Property 1.** The top-trading cycle algorithm outputs a matching that is Pareto optimal (not Pareto dominated by another matching).
2. **Property 2.** The top-trading cycle algorithm is strategyproof.

# 3 Stable Matchings

Consider two-sided stable matching market with $n$ candidates $C$ and $n$ jobs $J$. Each agent (candidate or job) has a preference list $P_i$ ranking their top $d$ choices (we assume that rankings are strict). If a candidate $c$ is not included in the preference list of a job $j$, then $j$ prefers to be unmatched than be matched to $c$. Similarly, if a job $j$ is not included on a candidate $c$'s preference list, than $c$ prefers to be unmatched than be matched to job $j$.

The goal is to find a perfect matching $M$ that is *stable*. We abuse notation a bit and let $M(c)$ denote a candidate $c$'s match in $M$ and $M(j)$ denote job $j$'s match in $M$.

**Definition 3.** (Stable Matching) A matching $M$ is stable if it does not have any blocking pairs. A pair $(c, j) \notin M$ forms a blocking pair with respect to $M$, if $c$ prefers $j$ to $M(j)$ and $j$ prefers $c$ to $M(c)$.

## 3.1 Deferred Acceptanace Algorithm

Turns out a stable matching always exists and can be computed by Gale Shapley's deferred acceptance (DA) algorithm. Algorithm 1 generalizes the classic DA algorithm for partial preferences. If each candidate has $d$ jobs on their preference list where $d < n$, then in a candidate-proposing deferred acceptance algorithm, if a candidate is rejected by all $d$ jobs on their list then they remain unmatched. Notice that the CPDA algorithm terminates when all candidates are either matched or have exhausted their preference list. Moreover, all candidates in $\mathcal{R}$ at the end of Algorithm 1 remain unmatched.

## 3.2 Properties of Deferred Acceptance

**Theorem 2.** The deferred acceptance algorithm always outputs a stable matching.

**Algorithm 1:** Candidate Proposing Deferred-Acceptance (CPDA) Algorithm

Fix an ordering over $\mathcal{C}$; Initialize $\mathcal{U} \leftarrow \mathcal{C}$ and $\mathcal{R} \leftarrow \emptyset$ and $\mu(j) \leftarrow \emptyset$ for all $j \in \mathcal{J}$
**while** $\mathcal{U} \setminus \mathcal{R} \neq \emptyset$ **do**
  Pick a candidate $c \in \mathcal{U}$ with the lowest index
  **if** $c \notin \mathcal{R}$ **then**
    Let $j$ be the most preferred job on $c$'s list that $c$ has not yet proposed to
    **if** $j$ is the last job on $c$'s list **then**
      add $c$ to $\mathcal{R}$
    **end**
    $c$ *proposes* to $j$; Let $c' = \mu(j)$
    **if** $j$ prefers $c$ to $c'$ **then**
      $j$ *rejects* $c'$
      **if** $c' \neq \emptyset$ **then**
        add $c'$ to $\mathcal{U}$
      **end**
      $j$ *accepts* $c$;
      Set $\mu(j) = c$ and remove $c$ from $\mathcal{U}$
    **end**
  **end**
**end**

**Proof.** (By contradiction) Let $M$ be the resulting matching. Suppose there exists a blocking pair $(h, s)$ such that $(h, s'), (h', s) \in M$. That is, $h$ prefers $s$ over $s'$, and $s$ prefers $h$ over $h'$. Thus $h$ must have offered to $s$ before $s'$. Either $s$ broke the match to $h$ at some point for some $h''$, or $s$ already had a match $h''$ that $s$ preferred over $h$. But students always trade up, so $s$ must prefer final match $h'$ over $h''$, which they prefer over $h$, which is a contradiction. □

Turns out that the output of candidate-proposing deferred acceptance (CPDA) is optimal for candidates and pessimal for jobs. That is, candidates are matched to their best achievable stable partner and jobs are matched to their worst-achievable stable partner. We prove this property next.

**Definition 4.** (Best and worst achievable stable partners) Let $I$ be an instance of the stable matching problem. Let $\mathcal{M}$ denote the set of all stable matchings on $I$. Define $\textbf{best}(c)$ to be most-preferred stable partner of $c$ among all matchings in $\mathcal{M}$. Define $\textbf{worst}(j)$ be the least-preferred stable partner of $j$ among all matchings in $\mathcal{M}$.

**Theorem 3.** The candidate-proposing DA algorithm matches each candidate $c$ to best($c$).

**Proof.** Suppose $k$ is the first round where a hospital $h$ is rejected by $s^* = \text{best}(h)$. This implies that $s^*$ received an offer from another hospital $h'$ that it prefers to $h$. Then, $s^* = \text{best}(h')$. Suppose not, suppose $s' = \text{best}(h')$, then since $h'$ proposed to $s^*$ by round $k$, $h'$ must have proposed to $s'$ before round $k$ and already been rejected. This contradicts that $k$ is the first round where a hospital $h$ is the first hospital to be rejected by best($h$).

Now suppose $M$ is the stable matching such that $(h, s^*) \in M$. Then, $(s, h')$ is a blocking pair with respect to $M$ which is a contradiction. □

**Theorem 4.** The candidate-proposing DA algorithm matches each job $j$ to worst($j$).

**Proof.** Similar to the above and is omitted. □

Stability has a trade-off. Stable matchings are not Pareto efficient among all possible matchings.

**Observation.** Stable matchings are not Pareto efficient.

**Proof.** Consider the following example.

| | | | |
|---|---|---|---|
| $s_1$ | $t_2$ | $t_1$ | $t_3$ |
| $s_2$ | $t_1$ | $t_2$ | $t_3$ |
| $s_3$ | $t_1$ | $t_2$ | $t_3$ |

| | | | |
|---|---|---|---|
| $t_1$ | $s_1$ | $s_3$ | $s_2$ |
| $t_2$ | $s_2$ | $s_1$ | $s_3$ |
| $t_3$ | $s_2$ | $s_1$ | $s_3$ |

The student-proposing DA matching will be $(s_1, t_1), (s_2, t_2), (s_3, t_3)$. The matching $(s_1, t_2), (s_2, t_1), (s_3, t_3)$ Pareto dominates the student-proposing matching. Because $s_1$ prefers $t_2$ over $t_1$ and $s_2$ prefers $t_1$ over $t_2$ and $s_3$ hasn't changed, this matching is strictly better for some students and as good for all other students.

□

However, within the space of stable matchings, it is not possible for one stable matching to Pareto dominate another one. This is because if one candidate is better off in another matching then this means that some job is worse off.

**Theorem 5.** Let $M$ be the output of the candidate-proposing deferred acceptance algorithm on instance $I$, then $M$ is not Pareto dominated by any other stable matching on $I$. This means that there does not exist any other stable matching $M'$ on $I$ such that:

- All candidates (or jobs) weakly prefer $M'$ to $M$, and
- At least one candidate (or job) strictly prefers $M'$ to $M$.

**Proof.** Suppose there exists a candidate $c$ that strictly prefers $M'$ to $M$. Then, by definition of stability to $M$, $j = M'(c)$ must strictly prefer $M$ to $M'$ (otherwise they would form a blocking pair wrt to $M$). Thus, $M'$ does not Pareto dominate $M$. □

## 3.3 Properties of Stable Matchings

**Lone Wolf Theorem.** When incomplete preferences are allowed, stable matchings admit a somewhat surprising property which is referred to as the *Lone Wolf Theorem* in the literature.

This property says that the set of matched agents are the same in *any* stable matching. That is, no matter what algorithm to used to generate the matching, the people who are unmatched in one are unmatched in all and those who are matched in one are matched in all of them.

**Theorem 6.** Let $I$ be an instance of the stable matching problem with incomplete preference list and $\mu$ be some stable matching of $I$. Prove that if a candidate $c$ is unmatched in $\mu$, then $c$ must be unmatched in every stable matching of $I$.

**Proof.** We first prove a useful property about the size of stable matchings.

**Claim.** Let $k$ be the number of candidate-job pairs that are matched in the output matching $\mu^*$ of CPDA on instance $I$. Then, exactly $k$ candidate-job pairs are matched in any stable matching $\mu$.

**Proof of Claim.** Consider a candidate $c$ that is not matched in $\mu*$, then $c$ must be unmatched in any stable matching because $\mu^*$ matches each candidate to their best achievable stable partner. Just since $n - k$ candidates are unmatched in $\mu$, they must be at least $n - k$ unmatched candidates in any stable matching $\mu$. Moreover, consider a job $j$ that is matched in $\mu^*$. Since $\mu^*$ matches each job to their worst achievable stable partner, $j$ must be matched in any stable matching. Thus, in any stable matching $\mu$, the number of

jobs that are matched should include the $k$ jobs that are matched in $\mu^*$. Thus, number of candidates that must be unmatched in $\mu$ is at least $n-k$, thus the number of matched candidates is at most $k$. Also, the number of jobs that are matched in $\mu$ is at least $k$. Since the number of matched candidates equals the number of matched jobs, there are exactly $k$ matched candidate, job pairs in $\mu$. □

We use the claim to prove the Lone Wolf Theorem. Consider a candidate $c$ that is unmatched in a stable matching $\mu$ and matched in another stable matching $\mu'$. Since $c$ is matched in $\mu'$ it must be matched in $\mu^*$. Since the number of matched candidates is the same in $\mu$ and $\mu^*$ and $c$ goes from being unmatched to matched, there exists a candidate $c'$ that was matched in $\mu$ that is now matched in $\mu^*$. Thus is a contradiction to the fact that $c'$ must get their best achievable matched in $\mu^*$, which should be at least as good as $\mu(c')$. □

**Strategyproofness of CPDA.** The lone-wolf theorem is useful in proving that the CPDA algorithm is dominant-strategyproof for the candidates.

**Theorem 7.** The candidate-proposing deferred acceptance algorithm is dominant-strategyproof for the candidates.

**Proof.** Suppose that the CPDA algorithm is not dominant-strataegyproof for the candidates. Then, there exists at least one candidate $c$ that can benefit from misreporting, keeping the preferences of all others fixed. More formally, consider an instance $I$ and a candidate $c$ with a true preference $P_c$ and another instance $I^{(1)}$ which is identical to $I$ except $c$'s preference list $P'_c \neq P_c$. Let $\mu$ and $\mu'$ be the matching output by CPDA on instances $I$ and $I^{(1)}$ respectively. Suppose $c$ prefers their match $j' = \mu'(c)$ to their match $j = \mu(c)$. This proof proceeds in several parts, outlined below.

(a) Consider a new instance $I^{(2)}$ which is identical to $I$ except $c$'s preference list now only consists of a single job $j'$. The matching $\mu'$ output when CPDA is run on $I^{(1)}$ continues to be a stable matching with respect to the instance $I^{(2)}$.

(b) Consider a new instance $I^{(3)}$ which is identical to $I$ except $c$'s truthful preference list $P_c$ is truncated at job $j'$. That is, all jobs below $j'$ are removed from $c$'s list. The matching $\mu''$ output when CPDA algorithm on $I^{(3)}$ must leave $c$ unmatched.

(c) Notice that the matching $\mu''$ output when CPDA algorithm on $I^{(3)}$ is also stable with respect to the instance $I^{(2)}$. Since $\mu''$ and $\mu'$ are both stable matchings wrt $I^{(2)}$ and $c$ is unmatched in $\mu''$ and matched in $\mu'$, this violates the lone wolf theorem. This is a contradiction and thus CPDA must be dominant-strategyproof for the candidates.

□

**Stategyproof and Stability.** No algorithm for two-sided matchings can be both stable and strategyproof (for both sides of the market). We proved this property for the case $n = 2$ on the assignment, but it holds more generally.

## 4 Voting

We can define a voting problem as consisting of a set of alternatives (or candidates) $A$ and a set $n$ of voters $V$, such that each voter $i$ in $V$ has a complete preference list $L_i$ that strictly ranks all alternatives in $A$. A voting rule is a function $f : (L_1, \ldots, L_n) \to A$ that takes the preferences lists as input and outputs an alternative from $A$ that is a *winner*. A voting rule is also called a *social-choice function*. We considered three popular voting rules in class.

- **Plularity** voting rule selects the winner with the most first-place votes.
- **Ranked-choice** voting rules proceeds as follows: if there is a candidate that has majority of first place votes, they win. Otherwise, eliminate the candidate $c$ with the least number of first-place votes from each voters list and all candidates below $c$ move up by one in that list. Repeat until a majority winner is found (which is always gauranteed when $n = 2$).
- **Borda score** voting rule assigns each candidate a score of $n - i + 1$ of being ranked in position $i$ of a voter's list. The candidate with the highest Borda score is the winner.

**Properties of Voting Rules.** We defined several properties of voting rules that are useful in analyzing them.

**Definition 5.** (Properties of Voting Rules)

- A voting rule is a Condorcet consistent if it always outputs the Condorcet winner (if they exist). A Condorcet winner is a candidate who *beats* every other candidate in a head-to-head comparison. In particular, a candidate $a$ beats a candidate $b$ in a head-to-head comparison if the number of voters that prefer $a$ to $b$ exceeds the number of voters that prefer $b$ to $a$. Plurality, ranked-choice and Borda are not Condorcet consistent. We proved in Assignment 4 that Schulze rule is Condorcet consistent.
- A voting rule $f$ is *onto* iff for every alternative $a \in A$, there exists some input $I = (L_1, \ldots, L_n)$ such that $a$ is the winner $f(I) = a$.
- A voting rule $f$ is *dictatorial* iff there exists a voter $i$ such that $f$ always outputs the

candidate that $i$ ranks first, regardless of the preferences of other voters.

- A voting rule $f$ is *monotone* iff $f(I) = a$ and for all input $I'$ such that for all voters, any candidate who was ranked below $a$ in $I$ is still ranked below $a$ in $I'$, then $f(I') = a$.
- A voting rule $f$ is *unanimous* iff whenever there exists two candidates $a$ and $b$ in input $I$ such that all voters prefer $a$ to $b$ then $f(I) \neq b$.

The main theorem we proved about voting rules is the Gibbard-Satterthwaite Theorem.

**Theorem 8.** (Gibbard-Satterthwaite) When there are 3 or more alternatives, a voting rule is strategyproof and onto if and only if it is dictatorial.

We proved this in three parts, outlined below.

- A voting rule is strategyproof if and only if it is monotone.
- If a voting rule is monotone and onto then it is unanimous.
- If a voting rule is monotone and unanimous then it is dictatorial.

## 5 Cake Cutting: Fair Division

Let the cake be the unit interval $[0, 1]$. Each player $i$ has a valuation function $v_i$: the value $v_i(S)$ for any subset $S$. We make the following assumptions about valuations:

- Assume $v_i$ is normalized with $v_i([0, 1]) = 1$.
- $v_i$ is additive on disjoint subsets: $v_i(A) + v_i(B) = v_i(A \cup B)$.

There are two notions of fairness, defined below.

**Definition 6.** ( **Proportional.**) An allocation $A_1, \ldots, A_n$ of cake to $n$ players is proportional if

$$v_i(A_i) \geq \frac{1}{n} \quad \text{for every player } i.$$

(**Envy free.**) An allocation $A_1, \ldots, A_n$ of cake to $n$ players is envy free if

$$v_i(A_i) \geq v_i(A_j) \quad \text{for every pair } i, j \text{ of players.}$$

**Lemma 3.** If an allocation is envy-free, it must be proportional.

**Proof.** Suppose the allocation $(A_1, \ldots, A_n)$ is not proportional. Then there exists an agent $i$ such that $v_i(A_i) < 1/n$. Since $v_i([0,1]) = 1$ and $\sum_i A_i = [0,1]$, this means there exists an agent $j$ who receives allocation $A_j$ such that $v_i(A_j) > 1/n$. Thus, agent $i$ envies agent $j$. □

## 5.1 Cut and Choose

## 5.2 Dubiens Spanier Protocol

## 5.3 Selfridge Conway