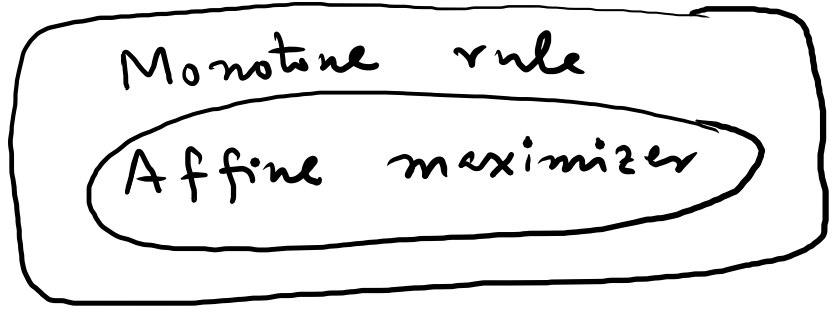
Recall:

Quasi-linear environment: DSIC allocation rules are the affine maximizers (Grove's Theorem and Roberts' Theorem)

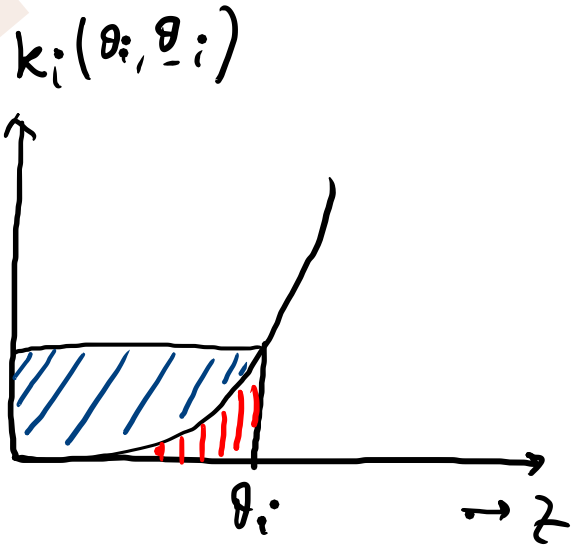
Single-Parameter domain: DSIC allocation rules are the monotone rules (Myerson's Lemma).

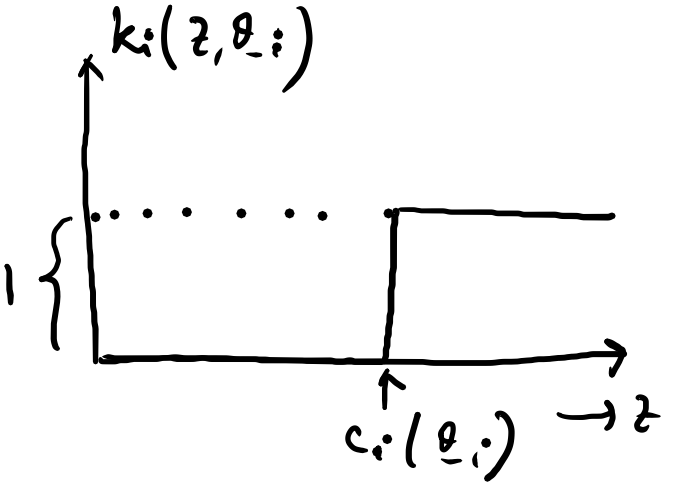


$$k(\cdot) = \left(k_1(\cdot), \ldots, k_n(\cdot)\right) \quad \text{monotone}$$

$$t_i(\theta) = \int_0^{\theta_i} z \, \frac{d}{dz}\left(k_i(z, \underline{\theta}_{-i})\right) \cdot dz$$

$$= \theta_i \cdot k_i(\theta_i, \underline{\theta}_{-i}) - \int_0^{\theta_i} k_i(z, \underline{\theta}_{-i}) \, dz$$





Implementability in Intermediate Domain

Convex domain: Each Θ_i is a convex set in some Euclidean space.

Weak monotonicity:- An allocation rule $k^*: \Theta \to \mathcal{K}$ is called weakly monotone if we have the following for $i \in [n]$, $\theta_i, \theta_i' \in \Theta_i$, $\underline{\theta}_i \in \Theta_{-i}$,

$$k^*(\theta_i, \underline{\theta}_i) = x \neq y = k^*(\theta_i', \underline{\theta}_i)$$

$$v_i(x, \theta_i) - v_i(y, \theta_i) \geq v_i(x, \theta_i') - v_i(y, \theta_i')$$

Theorem: If a mechanism $(k^*, t_1, \ldots, t_n)$ is DSIC, then k^* is weakly monotone. On the other hand, if Θ_i is a convex set for each $i \in [n]$, then, for every weakly monotone allocation rule $k^*(\cdot)$, there exist payment rules $t_1(\cdot), \ldots, t_n(\cdot)$ such that the mechanism $(k^*, t_1, \ldots, t_n)$ is DSIC.

Proof: (first part) Suppose a mechanism $(k^*, t_1, \ldots, t_n)$ is DSIC.

Then we have the following: for every $i \in [n]$, $\theta_i, \theta_i' \in \Theta_i$, $\theta_{-i} \in \Theta_{-i}$ such that $k^*(\theta_i, \theta_{-i}), k^*(\theta_i', \theta_{-i}) \in R_i$, we have $t_i(\theta_i, \theta_{-i}) = t_i(\theta_i', \theta_{-i})$. Since the mechanism is DSIC, in the type profile (θ_i, θ_{-i}), player i does not benefit by reporting θ_i' instead of θ_i. So we have

$$u_i(x, \theta_i) \geq u_i(y, \theta_i)$$

$$\Rightarrow v_i(x, \theta_i) + t_i(\theta_i, \underline{\theta}_i) \geq v_i(y, \theta_i) + t_i(\theta_i', \underline{\theta}_i)$$

$$\Rightarrow v_i(x, \theta_i) - v_i(y, \theta_i) \geq 0 \qquad \left[\because t_i(\theta_i, \underline{\theta}_i) = t_i(\theta_i', \underline{\theta}_i)\right]$$

—(1)

Similarly, since the mechanism is DSIC, player i does not benefit in the type profile $(\theta_i', \underline{\theta}_i)$ by reporting θ_i instead by θ_i'.

$$\Rightarrow v_i(y, \theta_i') - v_i(x, \theta_i') \geq 0$$

$$\Rightarrow \quad v_i(x, \theta_i') - v_i(y, \theta_i') \leq 0$$

$$\Rightarrow \quad v_i(x, \theta_i') - v_i(y, \theta_i') \leq v_i(x, \theta_i) - v_i(y, \theta_i)$$

[from inequality (1)]

A Glimpse of Algorithmic Mechanism Design : Knapsack Allocation

Lecture 12.2

Recall the Knapsack problem:

Input: n items with weights $w_1, \ldots, w_n$ and valuations $v_1, \ldots, v_n$; a knapsack of size W.

Goal: Find $S \subseteq [n]$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i$ is maximized.

This problem is weakly NP-complete.

We assume each item corresponds to a player and the valuation is the type (i.e. private information) of the player.

Crucial observation: Each player's type is a single real number.

Set of all allocations:

$$\mathcal{K} = \left\{ (x_1, \ldots, x_n) \in \{0,1\}^n : \sum_{i=1}^{n} x_i \cdot w_i \leq W \right\}$$

$i \in [n]$,

$$\mathcal{R}_i = \{ (x_1, \ldots, x_n) \in \mathcal{R} \mid x_i = 1 \}$$

This is a single parameter domain.

Is the Knapsack allocation rule monotone?

Of course it is !!

Problem: Computing knapsack allocation is NP-complete.

Real-world example of Knapsack problem:

W minutes are allocated for showing Ads.

n potential advertisements competing for slot.

w_i is the duration of Ad i.

v_i is the valuation/utility of Ad i (known only to player i).

Two step design rule for algorithmic mechanism design.

(i) Assume DSIC for free.

(ii) Design an allocation rule which (i) can be computed in polynomial time, (ii) approximates our optimal objective value, (iii) monotone.

Greedy allocation rule for Knapsack:

Delete all items having weight more than W.

Sort the items by value/weight. Let us assume w.l.o.g by renaming that

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \frac{v_3}{w_3} \geq \cdots \geq \frac{v_n}{w_n}$$

Pick items till there is space. Let the set of items picked $S = \{1, 2, \ldots, i^*\}$ i.e. $\sum_{j=1}^{i^*} w_j \leq W$, $\sum_{j=1}^{i^*+1} w_j > W$.

Output S if $\sum_{j=1}^{i^*} v_j \geq v_{i^*+1}$; otherwise output i^*+1.

This allocation rule can be computed in poly-nomial time.

The allocation rule is clearly monotone.

Theorem: $k^{greedy}(.)$ has an approximation factor of $\frac{1}{2}$.

proof:

$$ALG \geq \max\left\{ \sum_{j=1}^{i^*} v_j \;,\; v_{i^*+1} \right\}$$

$$\geq \frac{\sum_{j=1}^{i^*+1} v_j}{2}$$

$$\geq \frac{OPT}{2}$$

□

Mechanism design with money: There is an allocation rule which we implement with the help of suitable payment rules.

Mechanism design without money: No money is allowed.

Ex: Stable matching, stable roommate, House allocation etc.

Stable Matching

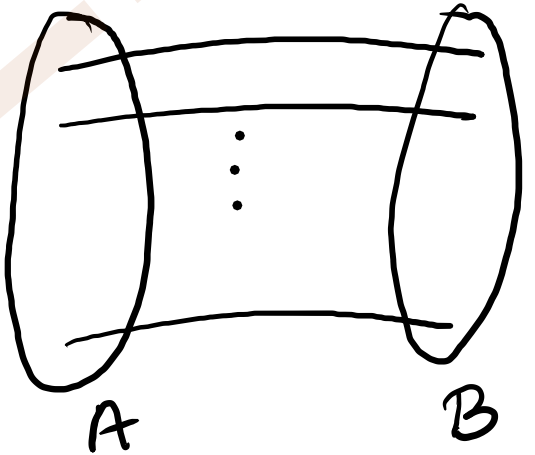
Setup: Set A of n men

set B of n women.

Each man has a preference which is a complete order over B.

Each woman has a preference which is a complete order over A.

Goal: Match each man with an woman so that the matching is "stable".



Blocking Pair: In a matching M of a stable matching instance, a pair $(a, b) \in A \times B$ is called a blocking pair if any of the following conditions hold.

(i) Both a and b are unmatched in M.

(ii) a is unmatched and $a \succ_b M(b)$, that is woman b prefers man a over her current partner in M.

(iii) b is unmatched and $b \succ_a M(a)$, that is man a prefers woman b over her current partner in M.

(iv) Both a and b are matched, $b \succ_a M(a)$ and $a \succ_b M(b)$, that is both a and b prefer each other over their current partners in M.

Stable matching: A matching having no blocking pair.

Gale-Shapley Theorem: Every stable matching instance has a stable matching. Moreover, one such stable matchings can be computed in polynomial time.

Proof: We present the Gale-Shapley algorithm.

Men-proposing deferred acceptance algorithm.

m_1 : $w_1 > w_2 > w_3$

m_2 : $w_2 > w_3 > w_1$

m_3 : $\underline{w_2} > \underline{w_1} > w_3$

w_1 : $m_2 > m_1 > m_3$

w_2 : $m_1 > m_2 > m_3$

w_3 : $m_3 > m_2 > m_1$

At the beginning all men and women are unmatched.

Iteration 1: m_1 proposes w_1.

current solution : $\{(m_1, w_1)\}$

Iteration 2: m_2 proposes w_2.

current solution : $\{(m_1, w_1), (m_2, w_2)\}$

Iteration 3: m_3 propose w_2

w_2 reject m_3 proposal since $m_2 \succ_{w_2} m_3$

current solution: $\{(m_1, w_1), (m_2, w_2)\}$

Iteration 4: m_3 propose w_1

w_1 reject m_3 since $m_1 \succ_{w_1} m_3$

current solution: $\{(m_1, w_1), (m_2, w_2)\}$

Iteration 5: m_3 propose w_3

current solution: $\{(m_1, w_1), (m_2, w_2), (m_3, w_3)\}$

Lecture 12.4

Gale-Shapley Theorem

Theorem: Every instance of the stable matching problem has a solution i.e. a stable matching.

Proof: Men-proposing deferred acceptance algorithm.

- Initially all men and women are unmatched.
- $M \leftarrow \phi$
- While there exists a man m who is unmatched and has not proposed all women {

- Let w be the most preferred woman of m whom he has not proposed yet.
- man m proposes woman w
- If w was unmatched then w accepts m's proposal. We have $M \leftarrow M \cup \{(m, w)\}$
- Else if w is matched with m' in M, but $m \succ_w m'$, then $M \leftarrow (M \setminus \{(m', w)\}) \cup \{(m, w)\}$

}

- Return m.

proof of correctness:- The while loop can terminate in two ways: (1) all men (and thus all women) are matched

(2) there exists a man who is unmatched, but he has proposed to all the women.

why while loop can not terminate because of (2)?

Observe: Any woman who has received a proposal, always remains matched in the successive iterations

of the algorithm although she can change her partner.

If there exists any man who has proposed to all women, then every woman has received at least one proposal. And thus all women are matched. But then all men are also matched.

The algorithm always outputs a perfect matching.

To prove: the output matching is stable.

That is there is no blocking pair.

For the sake of finding a contradiction, let us assume that there exists a blocking pair (m, w) for the output matching M.

$w \underset{m}{\geq} M(m) = w'$

$\boxed{m \underset{w}{\geq} M(w) = m'}$

The man m must have proposed w'. Since $w \underset{m}{\geq} w'$, the man m must have proposed woman w

in some iteration i. After i-th iteration the woman w remains matched. Moreover the partner of the woman w from $(i+1)$-th iteration is at least as preferable as m to the woman w. Hence, the final partner $M(w) = m'$ is more preferred to the woman w than m. But for (m, w) to be a blocking pair, we assumed $m \succ_w m'$. This is a contradiction ∎

Runtime: The while can iterate for $O(n(n-1)+1) = O(n^2)$

Properties of Stable Matching

Lecture 12.5

Men proposing deferred acceptance algorithm outputs the men-optimal stable matching.

For a man m, let us define $h(m)$ to be the woman whom m prefers most among all his partners in all stable matching.

Theorem: In the matching M output by the men-proposing deferred acceptance algorithm, every man m is matched with $h(m)$.

Proof: $R_i = \{(a,b) : \text{the woman } b \text{ has rejected the man } a \text{ in the first } i \text{ iterations}\}$

For an instance, suppose the algorithm runs for k iterations.

Claim: $i \in \{0,1,\ldots,k\}$, for every $(a,b) \in R_i$, there is no

stable matching where man a is matched with the woman b.

Proof by induction on i.

Base case: $i = 0$, $R_0 = \phi$, the claimed statement is vacuously true.

Inductive step: Let us assume the statement for $R_0, R_1, \ldots, R_i$.

Suppose a man a proposes a woman b in the (i+1) - th iteration.

case I: b was unmatched and has accepted a

Then $R_{i+1} = R_i$ and the proof follows from induction hypothesis.

case II: b has rejected some man a' who could be a also.

$$R_{i+1} = R_i \cup \{(a', b)\}$$

We only need to prove that there is no stable matching containing (a', b).

sub-case 1: $a' \neq a$

Then we have $a \succ_b a'$

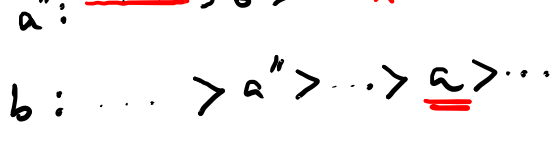
For the sake of finding a contradiction, let us assume that there exists a stable matching M' which contains (a', b)

$b \succ_a M'(a)$, $a \succ_b a'$

a: ——> b> ....

b: ...> a>...> a'> ....

(a, b) forms a blocking pair for M' which contradicts our assumption that M' is stable.



Subcase II: $a' = a$

If possible, suppose there exists a stable matching M' containing (a, b)

$b \succ_{a''} M'(a'')$ $\quad a'' \succ_b a = M'(b)$

(a'', b) forms a blocking pair for the matching

M' which contradicts our assumption that M' is stable.

Green book of stable matching 1989

Matching Manlove 207