Welfare Maximization

Yingkai Li

EC4501/EC4501HM Semester 2, AY2024/25

Mechanism Design

A mechanism design instance is denoted as $\Gamma_M = \big(N, \Omega, (v_i)_{i \in N}, (\Theta_i)_{i \in N}, F\big)$ where

- N is the set of agents;
- Ω is the set of outcomes;
- Θ_i is the set of agent i's "types" where $\theta_i \in \Theta_i$ is private information of i;
- $v_i : \Omega \times \Theta_i \to \mathbb{R}$ is agent i's value function;
- $F = F_1 \times \cdots \times F_n$ prior distribution over types.

Mechanism Design

A mechanism design instance is denoted as $\Gamma_M = \left(N, \Omega, (v_i)_{i \in N}, (\Theta_i)_{i \in N}, F\right)$ where

- N is the set of agents;
- Ω is the set of outcomes;
- Θ_i is the set of agent i's "types" where $\theta_i \in \Theta_i$ is private information of i;
- $v_i : \Omega \times \Theta_i \to \mathbb{R}$ is agent i's value function;
- $F = F_1 \times \cdots \times F_n$ prior distribution over types.

Let B_i be the report space of agent i.
A mechanism $M = (x, p)$:

- $x : B \to \Delta(\Omega)$;
- $p_i : B \to \mathbb{R}$, $\forall i$.

Smooth Auctions and Price of Anarchy

Smooth Auctions

Let $u_i(b;\theta_i) = v_i(x(b),\theta_i) - p_i(b)$ be the utility of agent i given bid profile b.
Let $\mathcal{R}(b) = \sum_i p_i(b)$.

Smooth Auctions

Let $u_i(b;\theta_i) = v_i(x(b),\theta_i) - p_i(b)$ be the utility of agent i given bid profile b.
Let $\mathcal{R}(b) = \sum_i p_i(b)$.

Definition (Smooth Auctions)

For parameters $\lambda \geq 0$ and $\mu \geq 1$, an auction is (λ,μ)-smooth if for every valuation profile $\mathbf{v} \in \mathcal{V}$ there exist bidding distributions $D_1^*(\mathbf{v}),\ldots,D_n^*(\mathbf{v})$ such that, for every bid profile b,

$$\sum_i \mathbb{E}_{b_i^* \sim D_i^*(\mathbf{v})}[u_i(b_i^*, b_{-i}; \mathbf{v}_i)] \geq \lambda \mathrm{Wel}(\mathbf{v}) - \mu \mathcal{R}(b).$$

Smooth Auctions

Let $u_i(b;\theta_i) = v_i(x(b),\theta_i) - p_i(b)$ be the utility of agent i given bid profile b.
Let $\mathcal{R}(b) = \sum_i p_i(b)$.

Definition (Smooth Auctions)

For parameters $\lambda \geq 0$ and $\mu \geq 1$, an auction is (λ, μ)-smooth if for every valuation profile $\mathbf{v} \in \mathcal{V}$ there exist bidding distributions $D_1^*(\mathbf{v}), \ldots, D_n^*(\mathbf{v})$ such that, for every bid profile b,

$$\sum_i \mathbb{E}_{b_i^* \sim D_i^*(\mathbf{v})}[u_i(b_i^*, b_{-i}; \mathbf{v}_i)] \geq \lambda \mathrm{Wel}(\mathbf{v}) - \mu \mathcal{R}(b).$$

First-price auction is $(\frac{1}{2}, 1)$-smooth.

- by bidding $\frac{v_i}{2}$, either wins and the utility is high, or loses and the total payment is high.

Smooth Auctions

Theorem

For any $\lambda \leq 1, \mu \geq 1$, if an auction is (λ, μ)-smooth, then for every product distribution F, every Bayes-Nash equilibrium of the auction has expected welfare at least $\frac{\lambda}{\mu} \cdot \mathrm{Wel}(F)$.

Same idea as in price of anarchy for first-price auctions!

Smooth Auctions

Theorem

For any $\lambda \leq 1, \mu \geq 1$, if an auction is (λ, μ)-smooth, then for every product distribution F, every Bayes-Nash equilibrium of the auction has expected welfare at least $\frac{\lambda}{\mu} \cdot \mathrm{Wel}(F)$.

Same idea as in price of anarchy for first-price auctions!

Intuition: utility in BNE $\geq$ utility given bidding strategy $D^* \geq \mathbf{E}[\lambda \mathrm{Wel}(\mathbf{v}) - \mu \mathcal{R}(b)]$.

Smooth Auctions

Theorem

For any $\lambda \leq 1, \mu \geq 1$, if an auction is (λ, μ)-smooth, then for every product distribution F, every Bayes-Nash equilibrium of the auction has expected welfare at least $\frac{\lambda}{\mu} \cdot \mathrm{Wel}(F)$.

Same idea as in price of anarchy for first-price auctions!

Intuition: utility in BNE $\geq$ utility given bidding strategy $D^* \geq \mathbf{E}[\lambda \mathrm{Wel}(\mathbf{v}) - \mu \mathcal{R}(b)]$.

welfare in BNE $=$ utility in BNE $+$ revenue $\geq \frac{\lambda}{\mu} \cdot \mathrm{Wel}(F)$.

Smooth Auctions

Theorem

For any $\lambda \leq 1, \mu \geq 1$, if an auction is (λ, μ)-smooth, then for every product distribution F, every Bayes-Nash equilibrium of the auction has expected welfare at least $\frac{\lambda}{\mu} \cdot \mathrm{Wel}(F)$.

Same idea as in price of anarchy for first-price auctions!

Intuition: utility in BNE $\geq$ utility given bidding strategy $D^* \geq \mathbf{E}[\lambda \mathrm{Wel}(\mathbf{v}) - \mu \mathcal{R}(b)]$.

welfare in BNE $=$ utility in BNE $+$ revenue $\geq \frac{\lambda}{\mu} \cdot \mathrm{Wel}(F)$.

Results apply to other auction formats: all-pay auction is $(\frac{1}{2}, 1)$-smooth. [assignment]

Composition Auctions

In practice, many auctions run in parallel.

- E.g., different sellers auction their products in parallel using first-price auctions.

Composition Auctions

In practice, many auctions run in parallel.

- E.g., different sellers auction their products in parallel using first-price auctions.

Is the equilibrium outcome approximately efficient in aggregation?

Composition Auctions

In practice, many auctions run in parallel.

- E.g., different sellers auction their products in parallel using first-price auctions.

Is the equilibrium outcome approximately efficient in aggregation?

Definition

A utility function is complement-free if there exists m additive valuations $f_1, \ldots, f_m$ such that for any set S,

$$f(s) = \max_{k \leq m} f_k(S).$$

Theorem (Composition Theorem)

If players have complement-free utility functions, then the simultaneous composition of (λ, μ)-smooth auctions is again a (λ, μ)-smooth auction.

Composition Auctions

In practice, many auctions run in parallel.

- E.g., different sellers auction their products in parallel using first-price auctions.

Is the equilibrium outcome approximately efficient in aggregation?

Definition

A utility function is complement-free if there exists m additive valuations $f_1, \ldots, f_m$ such that for any set S,

$$f(s) = \max_{k \leq m} f_k(S).$$

Theorem (Composition Theorem)

If players have complement-free utility functions, then the simultaneous composition of (λ, μ)-smooth auctions is again a (λ, μ)-smooth auction.

Corollary: PoA of the simultaneous composition of (λ, μ)-smooth auctions is at most $\frac{\mu}{\lambda}$.

Composition Auctions

Theorem (Composition Theorem)

If players have submodular utility functions, then the simultaneous composition of (λ, μ)-smooth auctions is again a (λ, μ)-smooth auction.

Composition Auctions

Theorem (Composition Theorem)

If players have submodular utility functions, then the simultaneous composition of (λ, μ)-smooth auctions is again a (λ, μ)-smooth auction.

Illustration for unit-demand auction and simultaneous first-price auction.

- given valuation profile v, find optimal allocation $x(v)$;
- consider strategy profile where each agent i only bids $\frac{v_{ij}}{2}$ in auction j where $x_{ij}(v) = 1$.

Composition Auctions

Theorem (Composition Theorem)

If players have submodular utility functions, then the simultaneous composition of (λ, μ)-smooth auctions is again a (λ, μ)-smooth auction.

Illustration for unit-demand auction and simultaneous first-price auction.

- given valuation profile v, find optimal allocation $x(v)$;
- consider strategy profile where each agent i only bids $\frac{v_{ij}}{2}$ in auction j where $x_{ij}(v) = 1$.

Reference: Roughgarden, T., Syrgkanis, V., & Tardos, E. (2017). The price of anarchy in auctions. Journal of Artificial Intelligence Research, 59, 59-101.

Efficiency and Polynomial-time Reduction

Revelation Mechanisms

A mechanism $M = (x, p)$ is a revelation mechanism if all agents are incentivized to report truthfully in mechanism M. I.e., $B_i = \Theta_i$ and

$$\mathbf{E}[v_i(x(\theta_i, \theta_{-i}), \theta_i) - p_i(\theta_i, \theta_{-i})] \geq \mathbf{E}[v_i(x(b_i, \theta_{-i}), \theta_i) - p_i(b_i, \theta_{-i})] \quad \forall i, \theta_i, b_i. \tag{IC}$$

$$\mathbf{E}[v_i(x(\theta_i, \theta_{-i}), \theta_i) - p_i(\theta_i, \theta_{-i})] \geq 0, \qquad \forall i, \theta_i. \tag{IR}$$

Revelation Mechanisms

A mechanism $M = (x, p)$ is a revelation mechanism if all agents are incentivized to report truthfully in mechanism M. I.e., $B_i = \Theta_i$ and

$$\mathbf{E}[v_i(x(\theta_i, \theta_{-i}), \theta_i) - p_i(\theta_i, \theta_{-i})] \geq \mathbf{E}[v_i(x(b_i, \theta_{-i}), \theta_i) - p_i(b_i, \theta_{-i})] \quad \forall i, \theta_i, b_i. \tag{IC}$$

$$\mathbf{E}[v_i(x(\theta_i, \theta_{-i}), \theta_i) - p_i(\theta_i, \theta_{-i})] \geq 0, \qquad \forall i, \theta_i. \tag{IR}$$

Lemma (Revelation Principle [Myerson '81])

It is without loss to focus on revelation mechanisms.

VCG Mechanisms

VCG mechanism: mechanism that implements efficient allocation in general environment.

- **allocation**: chooses outcome

$$\omega^* = \operatorname*{argmax}_{\omega \in \Omega} \sum_i v_i(\omega, \theta_i).$$

VCG Mechanisms

VCG mechanism: mechanism that implements efficient allocation in general environment.

- **allocation**: chooses outcome

$$\omega^* = \operatorname*{argmax}_{\omega \in \Omega} \sum_i v_i(\omega, \theta_i).$$

- **payment**: each agent i pays his externality on the welfare

$$p_i(\theta) = \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega, \theta_j) - \sum_{j \neq i} v_j(\omega^*, \theta_j) \geq 0.$$

VCG Mechanisms

VCG mechanism: mechanism that implements efficient allocation in general environment.

- **allocation**: chooses outcome

$$\omega^* = \operatorname*{argmax}_{\omega \in \Omega} \sum_i v_i(\omega, \theta_i).$$

- **payment**: each agent i pays his externality on the welfare

$$p_i(\theta) = \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega, \theta_j) - \sum_{j \neq i} v_j(\omega^*, \theta_j) \geq 0.$$

VCG mechanism is incentive compatible, individually rational, and maximizes social welfare.
VCG mechanism may not be implementable in polynomial time.

- specialize to second-price auction in single-item environment.

Welfare Maximization

Implementing the VCG mechanism requires solving the optimal allocation problem:

$$\omega^* = \operatorname*{argmax}_{\omega \in \Omega} \sum_i v_i(\omega, \theta_i).$$

Welfare Maximization

Implementing the VCG mechanism requires solving the optimal allocation problem:

$$\omega^* = \underset{\omega \in \Omega}{\operatorname{argmax}} \sum_i v_i(\omega, \theta_i).$$

Is this tractable in practice?

Welfare Maximization

Implementing the VCG mechanism requires solving the optimal allocation problem:

$$\omega^* = \operatorname*{argmax}_{\omega \in \Omega} \sum_i v_i(\omega, \theta_i).$$

Is this tractable in practice?

Example: (Knapsack problem) consider the allocation problem of servicing agents, where $\Omega \subseteq 2^N$.

- each agent has private value θ_i for being serviced;
- servicing each agent i requires a resource of r_i;
- there is a total budget of B on resource;
- allocation ω is feasible if and only if $\sum_{i \in \omega} r_i \leq B$.

Welfare Maximization

Implementing the VCG mechanism requires solving the optimal allocation problem:

$$\omega^* = \operatorname*{argmax}_{\omega \in \Omega} \sum_i v_i(\omega, \theta_i).$$

Is this tractable in practice?

Example: (Knapsack problem) consider the allocation problem of servicing agents, where $\Omega \subseteq 2^N$.

- each agent has private value θ_i for being serviced;
- servicing each agent i requires a resource of r_i;
- there is a total budget of B on resource;
- allocation ω is feasible if and only if $\sum_{i \in \omega} r_i \leq B$.

How to find the optimal allocation? Trying all combination requires time exponential in $|N|$. Not practical if $n = |N|$ is large!

Running Time

An algorithm is a polynomial-time algorithm if there exists $c \in (0, \infty)$ such that its running time $f(n)$ satisfies $f(n) = O(n^c)$.

Running Time

An algorithm is a polynomial-time algorithm if there exists $c \in (0, \infty)$ such that its running time $f(n)$ satisfies $f(n) = O(n^c)$.

Enumerating all subsets of N is not a polynomial-time algorithm: $2^n = \omega(n^c)$ for any $c < \infty$.

Running Time

An algorithm is a polynomial-time algorithm if there exists $c \in (0, \infty)$ such that its running time $f(n)$ satisfies $f(n) = O(n^c)$.

Enumerating all subsets of N is not a polynomial-time algorithm: $2^n = \omega(n^c)$ for any $c < \infty$.

Under the assumption that P$\neq$NP, the knapsack problem does not have any polynomial-time algorithm.

- There exist polynomial-time algorithms for approximating the optimal solutions in various settings.

Example: Knapsack Problem

Knapsack Problem

Example: Knapsack Problem

Knapsack Problem

Greedy algorithm:

1. sort agents in decreasing order of value per resource $\frac{\theta_i}{r_i}$;
2. allocate to agents with highest $\frac{\theta_i}{r_i}$ until the budget runs out.

Example: Knapsack Problem

Knapsack Problem

Greedy algorithm:

1. sort agents in decreasing order of value per resource $\frac{\theta_i}{r_i}$;
2. allocate to agents with highest $\frac{\theta_i}{r_i}$ until the budget runs out.

Max-feasible-value: allocate to the agent with highest value θ_i subject to the feasibility constraint $r_i \leq B$.

Example: Knapsack Problem

Theorem

The maximum of greedy algorithm and max-feasible-value is a 2-approximation to the optimal value in the knapsack problem.

Intuitively, we want to allocate according to the ratio $\frac{\theta_i}{r_i}$.

Example: Knapsack Problem

Theorem

The maximum of greedy algorithm and max-feasible-value is a 2-approximation to the optimal value in the knapsack problem.

Intuitively, we want to allocate according to the ratio $\frac{\theta_i}{r_i}$.

Why greedy is not optimal?

- The first agent that cannot be added via greedy can have a large value (which may be even larger than the total value selected in greedy).

Example: Knapsack Problem

Theorem

The maximum of greedy algorithm and max-feasible-value is a 2-approximation to the optimal value in the knapsack problem.

Intuitively, we want to allocate according to the ratio $\frac{\theta_i}{r_i}$.

Why greedy is not optimal?

- The first agent that cannot be added via greedy can have a large value (which may be even larger than the total value selected in greedy).

That infeasible agent must have value at most max-feasible-value.

Example: Knapsack Problem

Theorem

The maximum of greedy algorithm and max-feasible-value is a 2-approximation to the optimal value in the knapsack problem.

Intuitively, we want to allocate according to the ratio $\frac{\theta_i}{r_i}$.

Why greedy is not optimal?

- The first agent that cannot be added via greedy can have a large value (which may be even larger than the total value selected in greedy).

That infeasible agent must have value at most max-feasible-value.

Optimal solution $\leq$ greedy $+$ value of first infeasible agent $\leq$ greedy $+$ max-feasible-value.

Example: Knapsack Problem

Theorem

The maximum of greedy algorithm and max-feasible-value is a 2-approximation to the optimal value in the knapsack problem.

Intuitively, we want to allocate according to the ratio $\frac{\theta_i}{r_i}$.

Why greedy is not optimal?

- The first agent that cannot be added via greedy can have a large value (which may be even larger than the total value selected in greedy).

That infeasible agent must have value at most max-feasible-value.

Optimal solution $\leq$ greedy $+$ value of first infeasible agent $\leq$ greedy $+$ max-feasible-value.
$\Rightarrow$ Optimal solution $\leq 2 \cdot \max\{$greedy $+$ max-feasible-value$\}$.

Example: 3D Matching

3D Matching: serving each agent requires two types of resources. N: agents; X: resource type 1; Y: resource type 2.

- $L = \{(i, x, y)\}$: the set of feasible ways to serve the agents;
- find the maximum number of agents that can be served simultaneously.

Example: 3D Matching

3D Matching: serving each agent requires two types of resources. N: agents; X: resource type 1; Y: resource type 2.

- $L = \{(i, x, y)\}$: the set of feasible ways to serve the agents;
- find the maximum number of agents that can be served simultaneously.

Under the assumption that P$\neq$NP, the 3D matching problem does not have any polynomial-time algorithm.

Example: 3D Matching

3D Matching: serving each agent requires two types of resources. N: agents; X: resource type 1; Y: resource type 2.

- $L = \{(i, x, y)\}$: the set of feasible ways to serve the agents;
- find the maximum number of agents that can be served simultaneously.

Under the assumption that P$\neq$NP, the 3D matching problem does not have any polynomial-time algorithm.

Theorem

The greedy algorithm for finding the maximal matching is a 3-approximation to the optimal.

Intuition: in the greedy algorithm, when an agent is served, it will exclude at most two additional agents from the optimal matching.

Reduction from Algorithms to Mechanisms

The design of approximation algorithms does not take incentives of agents into consideration.

Reduction from Algorithms to Mechanisms

The design of approximation algorithms does not take incentives of agents into consideration.

Question: does there exist polynomial-time mechanism that guarantees good welfare approximations?

- A tricky issue: given an approximately optimal algorithm, there may not exist any mechanism that implements its allocation.

Reduction from Algorithms to Mechanisms

The design of approximation algorithms does not take incentives of agents into consideration.

Question: does there exist polynomial-time mechanism that guarantees good welfare approximations?

- A tricky issue: given an approximately optimal algorithm, there may not exist any mechanism that implements its allocation.

Theorem

For any $\beta \geq 1$ and given any polynomial time algorithm with approximation ratio β, there exists a polynomial time mechanism that achieves a β-approximation to the optimal welfare.

Reduction from Algorithms to Mechanisms

The design of approximation algorithms does not take incentives of agents into consideration.

Question: does there exist polynomial-time mechanism that guarantees good welfare approximations?

- A tricky issue: given an approximately optimal algorithm, there may not exist any mechanism that implements its allocation.

Theorem

For any $\beta \geq 1$ and given any polynomial time algorithm with approximation ratio β, there exists a polynomial time mechanism that achieves a β-approximation to the optimal welfare.

Idea: efficiency by matching.

- see illustration on board;
- apply efficiency in general equilibrium models to prove the reduction.

Reduction from Algorithms to Mechanisms

The design of approximation algorithms does not take incentives of agents into consideration.

Question: does there exist polynomial-time mechanism that guarantees good welfare approximations?

- A tricky issue: given an approximately optimal algorithm, there may not exist any mechanism that implements its allocation.

Theorem

For any $\beta \geq 1$ and given any polynomial time algorithm with approximation ratio β, there exists a polynomial time mechanism that achieves a β-approximation to the optimal welfare.

Idea: efficiency by matching.

- see illustration on board;
- apply efficiency in general equilibrium models to prove the reduction.

Reference: Hartline, J. D., Kleinberg, R., & Malekian, A. (2015). Bayesian incentive compatibility via matchings. Games and Economic Behavior, 92, 401-429.

General Equilibrium

Consider a market with n agents and n items.

- each agent i has unit value v_{ij} for item j;
- each agent i has demand at most f_i;
- each item j has supply at most g_j;
- $\sum_i f_i = \sum_j g_j = 1$.

General Equilibrium

Consider a market with n agents and n items.

- each agent i has unit value v_{ij} for item j;
- each agent i has demand at most f_i;
- each item j has supply at most g_j;
- $\sum_i f_i = \sum_j g_j = 1$.

Theorem

There exists a price p_j on each item j such that when each agent purchases their favorite consumption bundle,

- *the allocation is efficient;*
- *supply meets the demand, i.e., all items are sold out and all agents purchase up to their demand.*

General Equilibrium

Consider a market with n agents and n items.

- each agent i has unit value v_{ij} for item j;
- each agent i has demand at most f_i;
- each item j has supply at most g_j;
- $\sum_i f_i = \sum_j g_j = 1$.

Theorem

There exists a price p_j on each item j such that when each agent purchases their favorite consumption bundle,

- *the allocation is efficient;*
- *supply meets the demand, i.e., all items are sold out and all agents purchase up to their demand.*

Intuition: use tâtonnement rule to adjust the price

- gradually increase the price of the item with excessive demand.