Approximation in Mechanism Design

Aaron Roth

University of Pennsylvania

March 28 2024

Overview

1. Last lecture: how far can we go beyond the VCG mechanism when we want to optimize non-welfare objectives.

Overview

1. Last lecture: how far can we go beyond the VCG mechanism when we want to optimize non-welfare objectives.
2. This lecture: We grapple with computational complexity.

Overview

1. Last lecture: how far can we go beyond the VCG mechanism when we want to optimize non-welfare objectives.
2. This lecture: We grapple with computational complexity.
3. Recall the VCG mechanism must solve:

$$X(v) = \arg\max_{a \in A} \sum_i v_i(a)$$

Overview

1. Last lecture: how far can we go beyond the VCG mechanism when we want to optimize non-welfare objectives.
2. This lecture: We grapple with computational complexity.
3. Recall the VCG mechanism must solve:

$$X(v) = \arg\max_{a \in A} \sum_i v_i(a)$$

4. What do we do when this problem is hard to solve – e.g. NP-complete?

Approximation Algorithms

1. For many NP-complete problems we have good approximation algorithms — but this is not enough.

Approximation Algorithms

1. For many NP-complete problems we have good approximation algorithms — but this is not enough.
2. Recall: truthfulness and individual rationality of VCG depended on the choice rule being *exactly* welfare maximizing.

Approximation Algorithms

1. For many NP-complete problems we have good approximation algorithms — but this is not enough.
2. Recall: truthfulness and individual rationality of VCG depended on the choice rule being *exactly* welfare maximizing.
3. If we only find an alternative that achieves 99% of the optimal welfare, these guarantees break.

Approximation Algorithms

1. For many NP-complete problems we have good approximation algorithms — but this is not enough.
2. Recall: truthfulness and individual rationality of VCG depended on the choice rule being *exactly* welfare maximizing.
3. If we only find an alternative that achieves 99% of the optimal welfare, these guarantees break.
4. As a case study, we will consider *Knapsack auctions*.

Knapsack Auctions

Definition

In a *knapsack auction*:

- Each bidder $i \in \{1, \ldots, n\}$ has a public *size* $w_i \in \mathbb{R}_{\geq 0}$.

Knapsack Auctions

Definition

In a *knapsack auction*:

- Each bidder $i \in \{1, \dots, n\}$ has a public *size* $w_i \in \mathbb{R}_{\geq 0}$.
- The mechanism has a public *budget* $B \in \mathbb{R}_{\geq 0}$.

Knapsack Auctions

Definition

In a *knapsack auction*:

- Each bidder $i \in \{1, \ldots, n\}$ has a public *size* $w_i \in \mathbb{R}_{\geq 0}$.
- The mechanism has a public *budget* $B \in \mathbb{R}_{\geq 0}$.
- The *feasible alternatives* are all subsets of bidders of size $\leq B$:

$$A = \{S \subseteq \{1, \ldots, n\} : \sum_{i \in S} w_i \leq B\}$$

Knapsack Auctions

Definition

In a *knapsack auction*:

- Each bidder $i \in \{1, \ldots, n\}$ has a public *size* $w_i \in \mathbb{R}_{\geq 0}$.
- The mechanism has a public *budget* $B \in \mathbb{R}_{\geq 0}$.
- The *feasible alternatives* are all subsets of bidders of size $\leq B$:

$$A = \{S \subseteq \{1, \ldots, n\} : \sum_{i \in S} w_i \leq B\}$$

- For each $a \in A$ we write $a_i = 1$ if $i \in a$.

Knapsack Auctions

Definition

In a *knapsack auction*:

- Each bidder $i \in \{1, \ldots, n\}$ has a public *size* $w_i \in \mathbb{R}_{\geq 0}$.
- The mechanism has a public *budget* $B \in \mathbb{R}_{\geq 0}$.
- The *feasible alternatives* are all subsets of bidders of size $\leq B$:

$$A = \{S \subseteq \{1, \ldots, n\} : \sum_{i \in S} w_i \leq B\}$$

- For each $a \in A$ we write $a_i = 1$ if $i \in a$.
- These are single parameter domains. Each bidder i has a real value $v_i \in \mathbb{R}_{\geq 0}$, and their value for alternative a is $v_i \cdot a_i$

Knapsack Auctions

- Called a knapsack auction because solving:

$$\arg\max_{S \in A} \sum_{i \in S} v_i$$

is the NP-hard knapsack problem.

Knapsack Auctions

- Called a knapsack auction because solving:

$$\arg\max_{S \in A} \sum_{i \in S} v_i$$

  is the NP-hard knapsack problem.
- So: likely no polynomial time algorithm for this task.

Knapsack Auctions

- Called a knapsack auction because solving:

$$\arg\max_{S \in A} \sum_{i \in S} v_i$$

  is the NP-hard knapsack problem.
- So: likely no polynomial time algorithm for this task.
- A natural problem, modelling e.g. selling seats on an airplane to people who have different sized parties.

Knapsack Auctions

So what should we do?

- We could find a choice rule which *approximates* the social welfare objective and a pricing rule which makes it dominant strategy truthful.

Knapsack Auctions

So what should we do?

- We could find a choice rule which *approximates* the social welfare objective and a pricing rule which makes it dominant strategy truthful.
- We know that the only way to do this we have to find a *monotone non-decreasing* approximation algorithm.

Approximation

Definition

For a set of values and weights $v, w \in \mathbb{R}^n_{\geq 0}$, let:

$$\mathrm{OPT}(v, w) = \max_{S \subseteq [n] : \sum_{i \in S} w_i \leq B} \sum_{i \in S} v_i$$

Approximation

Definition

For a set of values and weights $v, w \in \mathbb{R}^n_{\geq 0}$, let:

$$\mathrm{OPT}(v, w) = \max_{S \subseteq [n] : \sum_{i \in S} w_i \leq B} \sum_{i \in S} v_i$$

A is an α-approximation algorithm for the Knapsack problem if for every $v, w \in \mathbb{R}^n_{\geq 0}$, $A(v, w) = S$ such that:

1. S is a feasible solution: $\sum_{i \in S} w_i \leq B$
2. S approximates OPT to within a factor of α: $\sum_{i \in S} v_i \geq \frac{\mathrm{OPT}(v,w)}{\alpha}$

Approximation

Definition

For a set of values and weights $v, w \in \mathbb{R}^n_{\geq 0}$, let:

$$\mathrm{OPT}(v, w) = \max_{S \subseteq [n] : \sum_{i \in S} w_i \leq B} \sum_{i \in S} v_i$$

A is an α-approximation algorithm for the Knapsack problem if for every $v, w \in \mathbb{R}^n_{\geq 0}$, $A(v, w) = S$ such that:

1. S is a feasible solution: $\sum_{i \in S} w_i \leq B$
2. S approximates OPT to within a factor of α: $\sum_{i \in S} v_i \geq \frac{\mathrm{OPT}(v,w)}{\alpha}$

Monotone Non Decreasing: for every $v, w \in \mathbb{R}^n_{\geq 0}$, and for every i and $v'_i > v_i$, if $S = A(v, w)$ and $S' = A((v'_i, v_{-i}), w)$, then:

$$i \in S \Rightarrow i \in S'.$$

Our Goal

- Goal: Come up with a monotone algorithm A that is also an α-approximation algorithm for the Knapsack problem.

Our Goal

- Goal: Come up with a monotone algorithm A that is also an α-approximation algorithm for the Knapsack problem.
- Observation: We can write the knapsack problem in the following integer linear optimization form:

$$\text{maximize} \sum_{i=1}^{n} x_i \cdot v_i$$

$$\text{such that:}$$

$$\sum_{i=1}^{n} x_i \cdot w_i \leq B$$

$$x_i \in \{0, 1\} \quad \forall i$$

A Complication

- Solving the integer program is NP hard. So...

A Complication

- Solving the integer program is NP hard. So...
- We are unlikely to be able to reason about structure of the optimal solution.

A Complication

- Solving the integer program is NP hard. So...
- We are unlikely to be able to reason about structure of the optimal solution.
- Instead, consider the following "relaxed" problem in which the x_i can be fractional:

$$\text{maximize} \sum_{i=1}^{n} x_i \cdot v_i$$

$$\text{such that:}$$

$$\sum_{i=1}^{n} x_i \cdot w_i \leq B$$

$$x_i \in [0,1] \quad \forall i$$

The Fractional Problem

- Write $\mathrm{OPT}_F(v, w)$ for the optimal value of this "fractional" problem.

The Fractional Problem

- Write $\mathrm{OPT}_F(v, w)$ for the optimal value of this "fractional" problem.
- Not the problem we want — but maybe we can understand its structure:

Lemma

For all $v, w \in \mathbb{R}^n_{\geq 0}$*:*

$$\mathrm{OPT}_F(v, w) \geq \mathrm{OPT}(v, w)$$

The Fractional Problem

- Write $\mathrm{OPT}_F(v, w)$ for the optimal value of this "fractional" problem.
- Not the problem we want — but maybe we can understand its structure:

Lemma

For all $v, w \in \mathbb{R}^n_{\geq 0}$:

$$\mathrm{OPT}_F(v, w) \geq \mathrm{OPT}(v, w)$$

Proof.

Any optimal solution to the integer version of the problem is a *feasible* solution to the fractional version, so $\mathrm{OPT}_F(v, w) \geq \mathrm{OPT}(v, w)$ □

Understanding the Fractional Problem

- If we can obtain an α-approximation to $\mathrm{OPT}_F(v, w)$ then we also get (at least!) an α-approximation to $\mathrm{OPT}(v, w)$.

Understanding the Fractional Problem

- If we can obtain an α-approximation to $\mathrm{OPT}_F(v, w)$ then we also get (at least!) an α-approximation to $\mathrm{OPT}(v, w)$.
- The fractional relaxation is simpler/easier to understand:

Lemma

Let x be a fractional solution obtaining value $\mathrm{OPT}_F(v, w)$ in the fractional knapsack problem. Let $i, j \in [n]$ be any pair of agents such that:

$$\frac{v_i}{w_i} > \frac{v_j}{w_j}.$$

Then $x_j > 0 \rightarrow x_i = 1$

Understanding the Fractional Problem

- Suppose otherwise: there is such an i, j pair with $x_j > 0$ but $x_i < 1$.

Understanding the Fractional Problem

- Suppose otherwise: there is such an i, j pair with $x_j > 0$ but $x_i < 1$.
- Define $\delta > 0$ by $\delta = \min((1 - x_i)\frac{w_i}{w_j}, x_j)$.

Understanding the Fractional Problem

- Suppose otherwise: there is such an i, j pair with $x_j > 0$ but $x_i < 1$.
- Define $\delta > 0$ by $\delta = \min((1 - x_i)\frac{w_i}{w_j}, x_j)$.
- Plan: Define a new solution x' and argue that it:

Understanding the Fractional Problem

- Suppose otherwise: there is such an i, j pair with $x_j > 0$ but $x_i < 1$.
- Define $\delta > 0$ by $\delta = \min((1 - x_i)\frac{w_i}{w_j}, x_j)$.
- Plan: Define a new solution x' and argue that it:
  1. Is feasible, and

Understanding the Fractional Problem

- Suppose otherwise: there is such an i, j pair with $x_j > 0$ but $x_i < 1$.
- Define $\delta > 0$ by $\delta = \min((1 - x_i)\frac{w_i}{w_j}, x_j)$.
- Plan: Define a new solution x' and argue that it:
  1. Is feasible, and
  2. Has higher objective value, contradicting the optimality of x.

Understanding the Fractional Problem

- Let $x'_\ell = x_\ell$ for all $\ell \notin \{i, j\}$, and let

  $$x'_j = x_j - \delta$$

  and

  $$x'_i = x_i + \delta \frac{w_j}{w_i}.$$

Understanding the Fractional Problem

- Let $x'_\ell = x_\ell$ for all $\ell \notin \{i, j\}$, and let

$$x'_j = x_j - \delta$$

and

$$x'_i = x_i + \delta \frac{w_j}{w_i}.$$

- Note that x' continues to satisfy the knapsack constraint: the change in size of the bundle was:

$$(\delta \frac{w_j}{w_i}) \cdot w_i - \delta w_j = \delta w_j - \delta w_j = 0$$

Understanding the Fractional Problem

- Let $x'_\ell = x_\ell$ for all $\ell \notin \{i, j\}$, and let

$$x'_j = x_j - \delta$$

and

$$x'_i = x_i + \delta \frac{w_j}{w_i}.$$

- Note that x' continues to satisfy the knapsack constraint: the change in size of the bundle was:

$$(\delta \frac{w_j}{w_i}) \cdot w_i - \delta w_j = \delta w_j - \delta w_j = 0$$

- By definition of δ: $x'_j \geq x_j - x_j = 0$ and $x'_i \leq x_i + ((1 - x_i)\frac{w_i}{w_j})\frac{w_j}{w_i} = 1$.

Understanding the Fractional Problem

- Let $x'_\ell = x_\ell$ for all $\ell \notin \{i, j\}$, and let

$$x'_j = x_j - \delta$$

  and

$$x'_i = x_i + \delta \frac{w_j}{w_i}.$$

- Note that x' continues to satisfy the knapsack constraint: the change in size of the bundle was:

$$(\delta \frac{w_j}{w_i}) \cdot w_i - \delta w_j = \delta w_j - \delta w_j = 0$$

- By definition of δ: $x'_j \geq x_j - x_j = 0$ and $x'_i \leq x_i + ((1 - x_i)\frac{w_i}{w_j})\frac{w_j}{w_i} = 1$.
- Hence (because x was feasible) x' is feasible.

Understanding the Fractional Problem

- The change in value of the bundle is:

$$(\delta \frac{w_j}{w_i}) \cdot v_i - \delta \cdot v_j > 0$$

Understanding the Fractional Problem

- The change in value of the bundle is:

$$(\delta \frac{w_j}{w_i}) \cdot v_i - \delta \cdot v_j > 0$$

- This follows because by assumption:

$$\frac{v_i}{w_i} > \frac{v_j}{w_j} \Rightarrow (\frac{w_j}{w_i}) \cdot v_i > v_j$$

Understanding the Fractional Problem

- The change in value of the bundle is:

$$(\delta \frac{w_j}{w_i}) \cdot v_i - \delta \cdot v_j > 0$$

- This follows because by assumption:

$$\frac{v_i}{w_i} > \frac{v_j}{w_j} \Rightarrow (\frac{w_j}{w_i}) \cdot v_i > v_j$$

- This contradicts the optimality of x.

Understanding the Fractional Problem

We can now give a simple combinatorial algorithm for the fractional version of the knapsack problem.

Understanding the Fractional Problem

We can now give a simple combinatorial algorithm for the fractional version of the knapsack problem.
Given our lemma, we know this algorithm must be optimal.
FractionalKnapsack(v, w):

Sort bidders in decreasing order by $\frac{v_i}{w_i}$ and set size $\leftarrow 0$ and $i \leftarrow 1$.
while size$+w_i \leq B$ **do**
  Set $x_i \leftarrow 1$, $\mathrm{size} \leftarrow \mathrm{size} + w_i$, $i \leftarrow i + 1$.
end while
Set $x_i \leftarrow \frac{B - \mathrm{size}}{w_i}$ and Set $x_j = 0$ for all $j > i$.
Return x.

The Integer Problem

- Can we use this algorithm to get a solution to the integer knapsack problem?

The Integer Problem

- Can we use this algorithm to get a solution to the integer knapsack problem?
- Note: Until the last step, the algorithm constructs an integer solution.

The Integer Problem

- Can we use this algorithm to get a solution to the integer knapsack problem?
- Note: Until the last step, the algorithm constructs an integer solution.
- What if we just remove the last step? How does this do?

The Integer Problem

- Can we use this algorithm to get a solution to the integer knapsack problem?
- Note: Until the last step, the algorithm constructs an integer solution.
- What if we just remove the last step? How does this do?
- Terribly! Consider the following example.

Example

We have two agents with $w_1 = v_1 = 10$ and $w_2 = 1$ and $v_2 = 1.1$. $B = 10$. Note that $\mathrm{OPT}(v, w) = 10$ However, $v_2/w_2 > v_1/w_1$, so the algorithm first picks agent 2, and then has no remaining space for agent 1. So the algorithm's solution has value only 1.1. We could extend this example to make the algorithm's solution arbitrarily worse!

The Integer Problem

- The problem: Leaving off the fractional portion of the solution may leave almost the entire knapsack empty.

The Integer Problem

- The problem: Leaving off the fractional portion of the solution may leave almost the entire knapsack empty.
- Lets try again. Note that WLOG, we can assume that for all i, $w_i \leq B$.
  Greedy2(v, w):

  Sort bidders in decreasing order by $\frac{v_i}{w_i}$ and set size $\leftarrow 0$ and $i \leftarrow 1$. Set $S \leftarrow \emptyset$.
  while size$+w_i \leq B$ **do**
  Set $S \leftarrow S \cup \{i\}$, $\text{size} \leftarrow \text{size} + w_i$, $i \leftarrow i + 1$.
  end while
  if $\sum_{j \in S} v_j \geq v_i$ **then**
  Output S.
  else
  Output $\{i^*\}$ where $i^* = \arg\max_i v_i$.
  end if

The Integer Problem

Theorem

Greedy2 achieves a 2-approximation algorithm for the Knapsack problem.

The Integer Problem

- By construction, for every agent j:

$$j \notin S \cup \{i\} \Rightarrow x_j^* = 0$$

where x^* is the optimal fractional solution to the fractional knapsack instance (v, w).

The Integer Problem

- By construction, for every agent j:

$$j \notin S \cup \{i\} \Rightarrow x_j^* = 0$$

  where x^* is the optimal fractional solution to the fractional knapsack instance (v, w).

- Hence:

$$\sum_{j \in S} v_j + v_i \geq \mathrm{OPT}_F(v, w) \geq \mathrm{OPT}(v, w).$$

The Integer Problem

- By construction, for every agent j:

$$j \notin S \cup \{i\} \Rightarrow x_j^* = 0$$

  where x^* is the optimal fractional solution to the fractional knapsack instance (v, w).

- Hence:

$$\sum_{j \in S} v_j + v_i \geq \mathrm{OPT}_F(v, w) \geq \mathrm{OPT}(v, w).$$

- Therefore:

$$\max(\sum_{j \in S} v_j, v_i) \geq \frac{OPT(v, w)}{2}$$

The Integer Problem

- By construction, for every agent j:

$$j \notin S \cup \{i\} \Rightarrow x_j^* = 0$$

where x^* is the optimal fractional solution to the fractional knapsack instance (v, w).

- Hence:

$$\sum_{j \in S} v_j + v_i \geq \mathrm{OPT}_F(v, w) \geq \mathrm{OPT}(v, w).$$

- Therefore:

$$\max(\sum_{j \in S} v_j, v_i) \geq \frac{OPT(v, w)}{2}$$

- And $v_{i^*} \geq v_i$ by definition. So:

$$\max(\sum_{i \in S} v_i, v_{i^*}) \geq \frac{OPT(v, w)}{2}$$

Establishing Truthfulness

Theorem

Greedy2 is monotone non-decreasing for every agent i.

Establishing Truthfulness

Theorem

Greedy2 is monotone non-decreasing for every agent i.

Hence, there is a dominant strategy truthful 2-approximation algorithm for the Knapsack Auction problem.

Establishing Truthfulness

- Fix any $w, v \in \mathbb{R}^n_{\geq 0}$, any agent i, and let $v'_i > v_i$. Write $v' = (v'_i, v_{-i})$. Let $T = Greedy2(v, w)$ and $T' = Greedy2(v', w)$.

Establishing Truthfulness

- Fix any $w, v \in \mathbb{R}^n_{\geq 0}$, any agent i, and let $v'_i > v_i$. Write $v' = (v'_i, v_{-i})$. Let $T = Greedy2(v, w)$ and $T' = Greedy2(v', w)$.
- To show: $i \in T \Rightarrow i \in T'$.

Establishing Truthfulness

- Fix any $w, v \in \mathbb{R}^n_{\geq 0}$, any agent i, and let $v'_i > v_i$. Write $v' = (v'_i, v_{-i})$. Let $T = \mathit{Greedy}2(v, w)$ and $T' = \mathit{Greedy}2(v', w)$.
- To show: $i \in T \Rightarrow i \in T'$.
- Write $S \doteq S(v, w)$ and $S' \doteq S(v', w)$ for the intermediate sets S generated by Greedy2 on each instance.

Establishing Truthfulness

- Fix any $w, v \in \mathbb{R}^n_{\geq 0}$, any agent i, and let $v'_i > v_i$. Write $v' = (v'_i, v_{-i})$. Let $T = Greedy2(v, w)$ and $T' = Greedy2(v', w)$.
- To show: $i \in T \Rightarrow i \in T'$.
- Write $S \doteq S(v, w)$ and $S' \doteq S(v', w)$ for the intermediate sets S generated by Greedy2 on each instance.
- First we argue:

$$i \in S \Rightarrow i \in S'$$

Establishing Truthfulness

- Note: S and S' represent the prefix of the bidders of total size $\leq B$ when sorted in decreasing order of $\frac{v_j}{w_j}$.

Establishing Truthfulness

- Note: S and S' represent the prefix of the bidders of total size $\leq B$ when sorted in decreasing order of $\frac{v_j}{w_j}$.
- When agent i increases his value from v_i to v_i' he can only move earlier in this sorted ordering.

Establishing Truthfulness

- Note: S and S' represent the prefix of the bidders of total size $\leq B$ when sorted in decreasing order of $\frac{v_j}{w_j}$.
- When agent i increases his value from v_i to v_i' he can only move earlier in this sorted ordering.
- So: if he was in the prefix S he is still in the prefix S'.

Establishing Truthfulness

- Note: S and S' represent the prefix of the bidders of total size $\leq B$ when sorted in decreasing order of $\frac{v_j}{w_j}$.
- When agent i increases his value from v_i to v_i' he can only move earlier in this sorted ordering.
- So: if he was in the prefix S he is still in the prefix S'.
- Hence: If $T = S$ and $T' = S'$, then on this instance, the algorithm is monotone.

Establishing Truthfulness

- Note also that if $i \in S$, then $\sum_{j \in S'} v'_j \geq \sum_{j \in S} v_j$. Hence, if $i \in S$, then if $T = S$, $T' = S$.

Establishing Truthfulness

- Note also that if $i \in S$, then $\sum_{j \in S'} v'_j \geq \sum_{j \in S} v_j$. Hence, if $i \in S$, then if $T = S$, $T' = S$.
- The other case: $i = i^*$ and $v_i > \sum_{j \in S} v_j$.

Establishing Truthfulness

- Note also that if $i \in S$, then $\sum_{j \in S'} v'_j \geq \sum_{j \in S} v_j$. Hence, if $i \in S$, then if $T = S$, $T' = S$.
- The other case: $i = i^*$ and $v_i > \sum_{j \in S} v_j$.
- Here we also have $i \in T'$. i remains the highest bidder, and so is either output as $T' = \{i^*\}$ or is output as $T' = S'$ with $i \in S'$.

Establishing Truthfulness

- Note also that if $i \in S$, then $\sum_{j \in S'} v'_j \geq \sum_{j \in S} v_j$. Hence, if $i \in S$, then if $T = S$, $T' = S$.
- The other case: $i = i^*$ and $v_i > \sum_{j \in S} v_j$.
- Here we also have $i \in T'$. i remains the highest bidder, and so is either output as $T' = \{i^*\}$ or is output as $T' = S'$ with $i \in S'$.
- So: we have shown that there exists a polynomial time 2-approximation for the Knapsack problem that makes truthful bidding a dominant strategy for all players.

Thanks!

See you next class — stay healthy!