Limits on Computationally Efficient VCG-Based Mechanisms for Combinatorial Auctions and Public Projects

Dave Buchfuhrer

CALIFORNIA INSTITUTE OF TECHNOLOGY

May 20, 2011

Outline

Outline

Combinatorial Auctions

A combinatorial auction consists of

- n players $1, \ldots, n$
- m items $1, \ldots, m$
- n valuation functions $v_1, \ldots, v_n$ where $v_i : 2^{[m]} \to \mathbb{R}_0{}^+$

An allocation is a partition of the items $S_1, \ldots, S_n$ where

- $S_i \cap S_j = \emptyset$ for $i \neq j$
- $\bigcup_i S_i \subseteq [m]$

We wish to maximize the social welfare, $\sum_i v_i(S_i)$.

Combinatorial Public Projects

A combinatorial public project consists of

- n players $1, \ldots, n$
- m items $1, \ldots, m$
- n valuation functions $v_1, \ldots, v_n$ where $v_i : 2^{[m]} \rightarrow {\mathbb{R}_0}^+$
- An integer k, $0 \leq k \leq m$.

An allocation is a subset $S \subseteq [m]$ of size k.
We wish to maximize the social welfare, $\sum_i v_i(S)$.

Truthful Mechanism

Definition (Truthful Mechanism)

A mechanism $\mathcal{M}$ consists of an allocation algorithm A and an algorithm to determine the prices $p_1, \ldots, p_n$ to charge the players. $\mathcal{M}$ is truthful if for any $v_1, \ldots, v_n$,

$$v_i(A(v_1, \ldots, v_n)) - p_i \geq v_i(A(v_1, \ldots, v_i', \ldots, v_n)) - p_i'$$

In other words, no player can possibly benefit by falsely reporting its valuation function.

Question

Are efficient truthful mechanisms capable of approximating the social welfare as well as other polynomial-time algorithms?

VCG Mechanism

Both problems can be solved truthfully by the VCG mechanism if a maximal-in-range algorithm is used.

Definition (Maximal-in-Range (MIR))

An allocation algorithm A takes in valuation functions and outputs an allocation. If R is the set of possible allocations output by A, A is maximal-in-range if it always outputs an allocation from R maximizing the social welfare.

We examine the capabilities of maximal-in-range mechanisms.

Outline

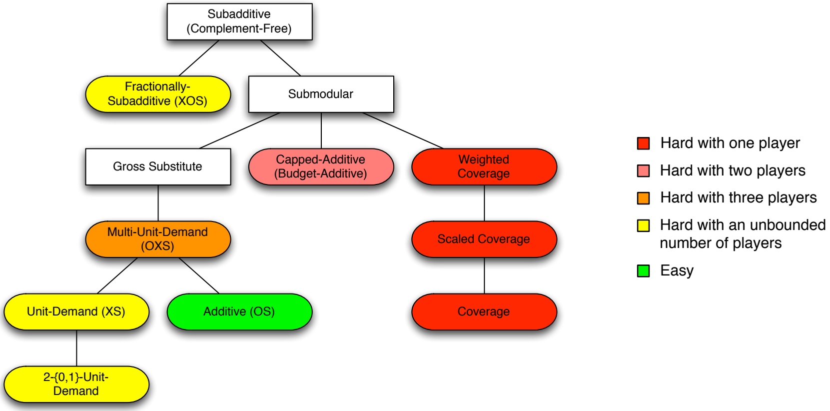
How to Show Hardness for MIR Algorithms

We use the following general framework to show that MIR algorithms are bad approximations.

1. Show that a good approximation ratio implies a large range
2. Show that a large range implies a large VC-dimension
3. Embed a reduction into the VC-dimension
4. *A* can't be MIR and poly-time unless $\mathrm{NP} \subseteq \mathrm{P}/\mathrm{poly}$

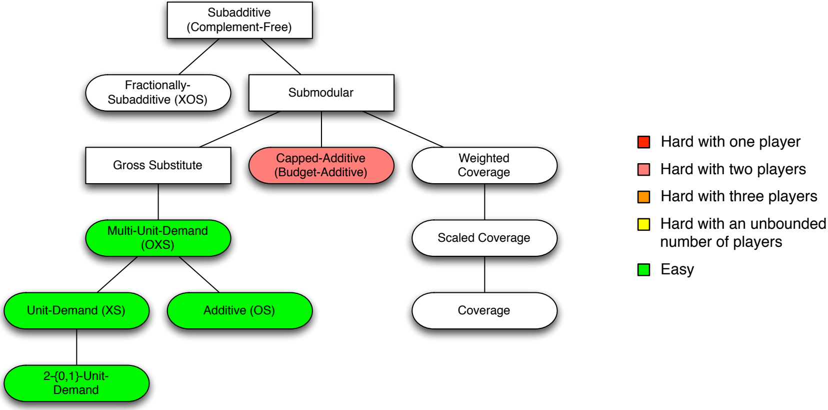
MIR Results - Combinatorial Public Projects

We showed that all public projects in the hierarchy (except additive) don't have better MIR approximations than $\sqrt{m}$ [BSS10], matching a $\sqrt{m}$ approximation in [SS08].



MIR Results - Combinatorial Auctions

We showed that capped-additive auctions are hard to approximate by MIR algorithms better than $\min(n, O(\sqrt{m}))$ [BDF$^+$10], matching a $\min(n, 2\sqrt{m})$ approximation in [DNS05].



Auctions are less amenable to study than public projects

Outline

Coverage Valuations

Definition (Coverage Valuation)

A coverage valuation v_i consists of sets $V_i^1, \ldots, V_i^m$ and the value of a set S is $v_i(S) = \left| \bigcup_{j \in S} V_i^j \right|$.

Coverage Valuations

Definition (Coverage Valuation)

A coverage valuation v_i consists of sets $V_i^1, \ldots, V_i^m$ and the value of a set S is $v_i(S) = \left|\bigcup_{j \in S} V_i^j\right|$.

Example (Exercise Machines)

- ☐ Arms
- ☐ Legs
- ☐ Chest
- ☐ Core
- ☐ Cardio

Coverage Valuations

Definition (Coverage Valuation)

A coverage valuation v_i consists of sets $V_i^1, \ldots, V_i^m$ and the value of a set S is $v_i(S) = \left| \bigcup_{j \in S} V_i^j \right|$.

Example (Exercise Machines)

- ☑ Arms
- ☐ Legs
- ☑ Chest
- ☐ Core
- ☐ Cardio

Coverage Valuations

Definition (Coverage Valuation)

A coverage valuation v_i consists of sets $V_i^1, \ldots, V_i^m$ and the value of a set S is $v_i(S) = \left| \bigcup_{j \in S} V_i^j \right|$.

Example (Exercise Machines)

- ☑ Arms
- ☐ Legs
- ☑ Chest
- ☐ Core
- ☐ Cardio

Coverage Valuations

Definition (Coverage Valuation)

A coverage valuation v_i consists of sets $V_i^1, \ldots, V_i^m$ and the value of a set S is $v_i(S) = \left| \bigcup_{j \in S} V_i^j \right|$.

Example (Exercise Machines)

- ☑ Arms
- ☐ Legs
- ☑ Chest
- ☐ Core
- ☑ Cardio

Cheating at Solitaire

Definition (Scaled Coverage)

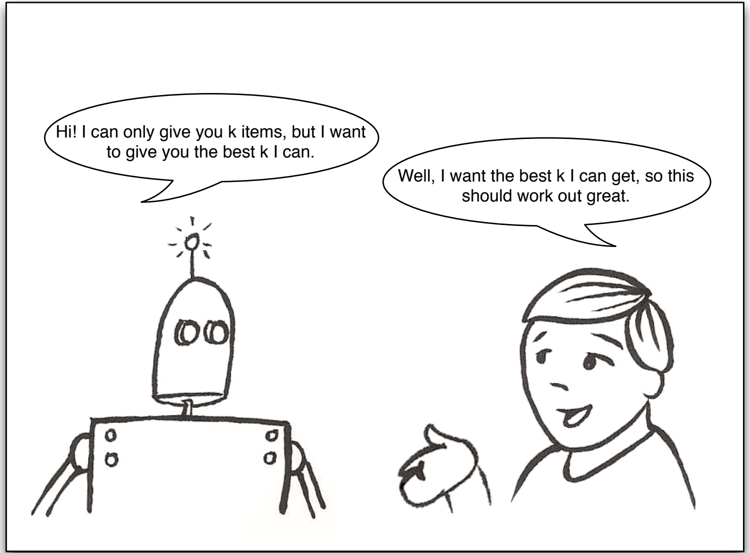
A scaled coverage valuation v_i consists of sets $V_i^1, \ldots, V_i^n$ and a scaling factor α. The value of a set S is

$$v_i(S) = \alpha \left| \bigcup_{j \in S} V_i^j \right| .$$

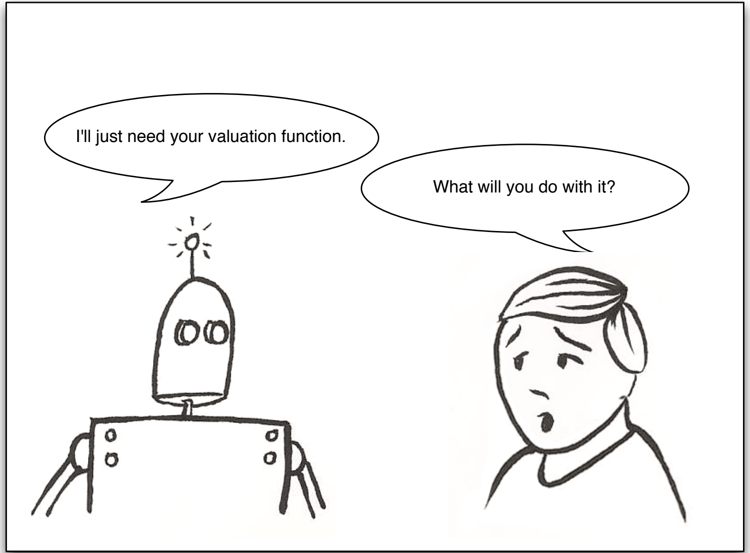
Theorem

Public projects with a single scaled coverage valuation player can't be approximated better than $\sqrt{m}$ by polynomial-time truthful mechanisms unless $NP \subseteq P/poly$.

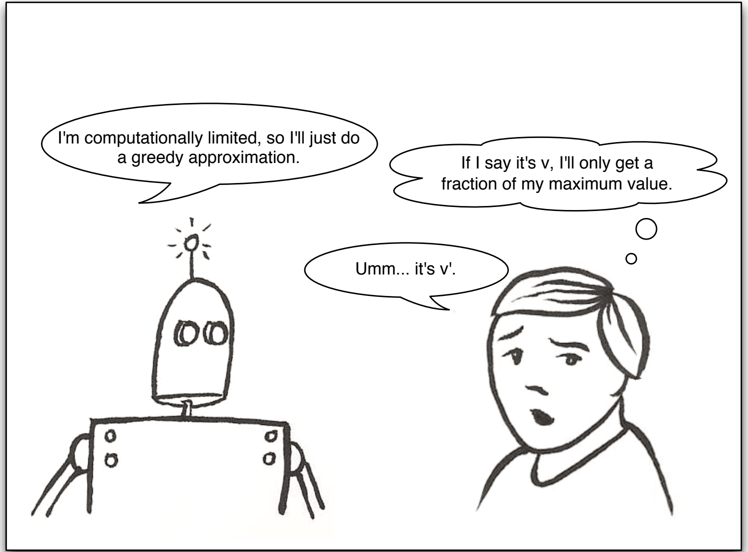
The Weird Scenario



The Weird Scenario



The Weird Scenario



Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Greedy Algorithm not Optimal)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1, 2\}, V^2 = \{3, 4\}, V^3 = \{5, 6\}, V^4 = \{1, 3, 5\}$

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Greedy Algorithm not Optimal)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1, 2\}$, $V^2 = \{3, 4\}$, $V^3 = \{5, 6\}$, $V^4 = \{1, 3, 5\}$
- If $k = 3$, $|V^1 \cup V^2 \cup V^3| = 6$, but greed gets value 5

Round	Item 1	Item 2	Item 3	Item 4	Covered Set
1	2	2	2	3	

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Greedy Algorithm not Optimal)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1,2\}, V^2 = \{3,4\}, V^3 = \{5,6\}, V^4 = \{1,3,5\}$
- If $k = 3$, $|V^1 \cup V^2 \cup V^3| = 6$, but greed gets value 5

Round	Item 1	Item 2	Item 3	Item 4	Covered Set
1	2	2	2	3	{1,3,5}

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Greedy Algorithm not Optimal)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1, 2\}, V^2 = \{3, 4\}, V^3 = \{5, 6\}, V^4 = \{1, 3, 5\}$
- If $k = 3$, $|V^1 \cup V^2 \cup V^3| = 6$, but greed gets value 5

Round	Item 1	Item 2	Item 3	Item 4	Covered Set
1	2	2	2	3	{1,3,5}
2	1	1	1	-	

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Greedy Algorithm not Optimal)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1, 2\}$, $V^2 = \{3, 4\}$, $V^3 = \{5, 6\}$, $V^4 = \{1, 3, 5\}$
- If $k = 3$, $|V^1 \cup V^2 \cup V^3| = 6$, but greed gets value 5

Round	Item 1	Item 2	Item 3	Item 4	Covered Set
1	2	2	2	3	{1,3,5}
2	1	1	1	-	{1,2,3,5}

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Greedy Algorithm not Optimal)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1, 2\}$, $V^2 = \{3, 4\}$, $V^3 = \{5, 6\}$, $V^4 = \{1, 3, 5\}$
- If $k = 3$, $|V^1 \cup V^2 \cup V^3| = 6$, but greed gets value 5

Round	Item 1	Item 2	Item 3	Item 4	Covered Set
1	2	2	2	3	{1,3,5}
2	1	1	1	-	{1,2,3,5}
3	-	1	1	-	

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Greedy Algorithm not Optimal)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1, 2\}, V^2 = \{3, 4\}, V^3 = \{5, 6\}, V^4 = \{1, 3, 5\}$
- If $k = 3$, $|V^1 \cup V^2 \cup V^3| = 6$, but greed gets value 5

Round	Item 1	Item 2	Item 3	Item 4	Covered Set
1	2	2	2	3	{1,3,5}
2	1	1	1	-	{1,2,3,5}
3	-	1	1	-	{1,2,3,4,5}

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Lies Improve Welfare)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1, 2\}, V^2 = \{3, 4\}, V^3 = \{5, 6\}, V^4 = \{1, 3, 5\}$
- Define v' by $V^{1'} = \{1\}, V^{2'} = \{2\}, V^{3'} = \{3\}, V^{4'} = \{\}$

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Lies Improve Welfare)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1,2\}, V^2 = \{3,4\}, V^3 = \{5,6\}, V^4 = \{1,3,5\}$
- Define v' by $V^{1'} = \{1\}, V^{2'} = \{2\}, V^{3'} = \{3\}, V^{4'} = \{\}$
- The greedy algorithm on v' chooses $1,2,3$

Why is the greedy algorithm not truthful?

Definition (Greedy Algorithm)

The greedy algorithm chooses k items by repeatedly choosing the item of maximum marginal value.

Example (Lies Improve Welfare)

- $v(S) = \left|\bigcup_{j \in S} V^j\right|$
- $V^1 = \{1,2\}$, $V^2 = \{3,4\}$, $V^3 = \{5,6\}$, $V^4 = \{1,3,5\}$
- Define v' by $V^{1'} = \{1\}$, $V^{2'} = \{2\}$, $V^{3'} = \{3\}$, $V^{4'} = \{\}$
- The greedy algorithm on v' chooses $1,2,3$
- If a player has value function v and declares v, he gets value 5
- If a player has value function v and declares v', he gets value 6

The Problem

The mechanism does the best it can to help the player out, but the player will still lie to it. Why does this happen?

- For efficiency, the mechanism must run in polynomial time
- For truthfulness, the players are not computationally limited

The Problem

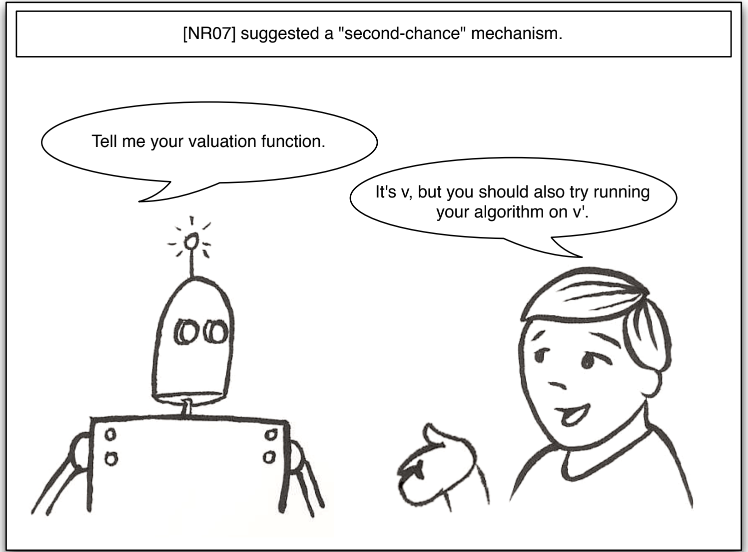
The mechanism does the best it can to help the player out, but the player will still lie to it. Why does this happen?

- For efficiency, the mechanism must run in polynomial time
- For truthfulness, the players are not computationally limited

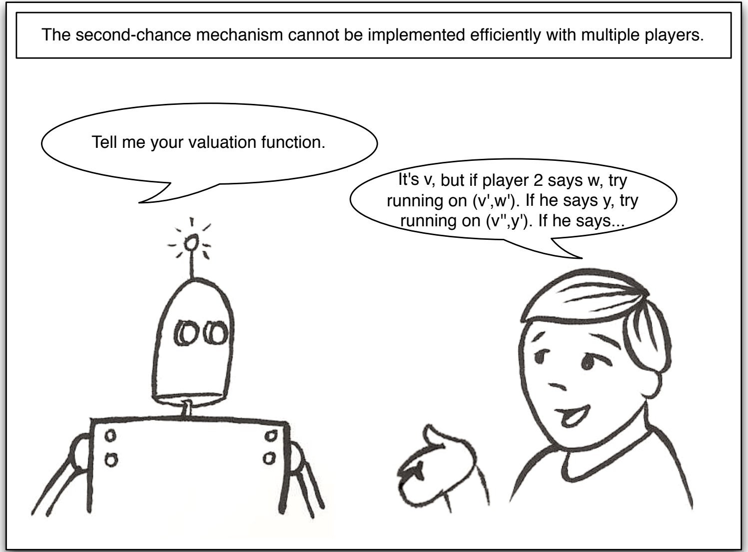
Asymmetry between efficiency and truthfulness is the problem here. We want to resolve this such that:

- Problems that should be easy are easy
- Problems that should be hard are hard

Second-Chance Mechanism

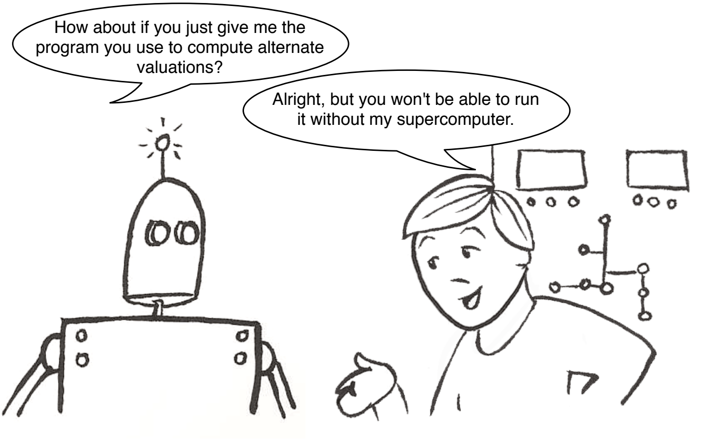


Second-Chance Mechanism

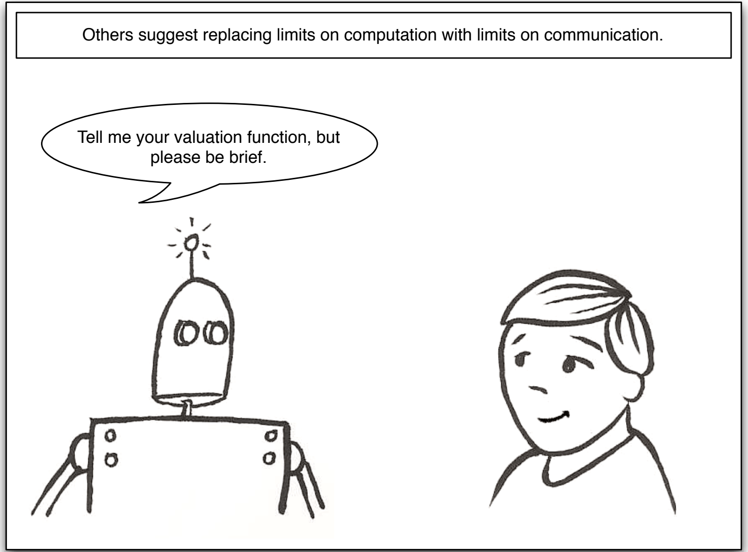


Second-Chance Mechanism

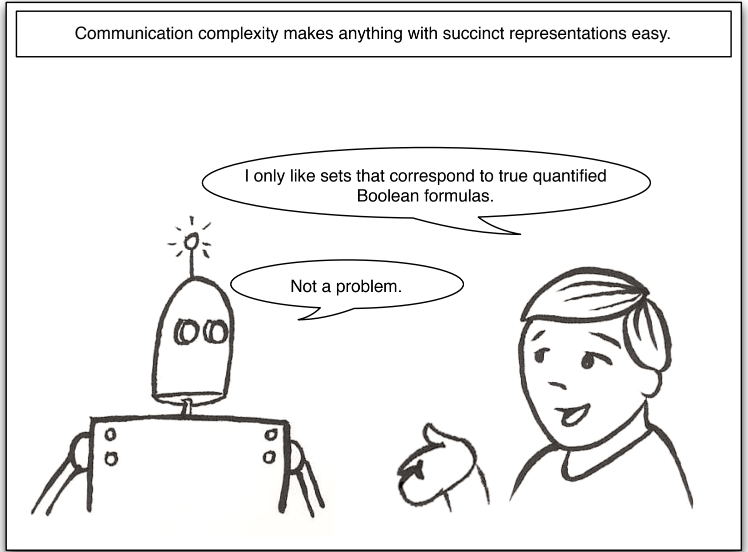
The second-chance mechanism cannot be efficiently implemented with multiple players.



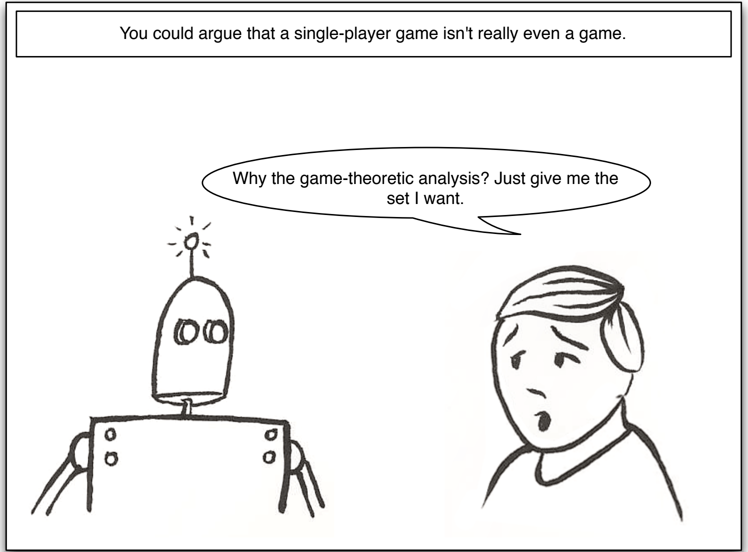
Communication Complexity



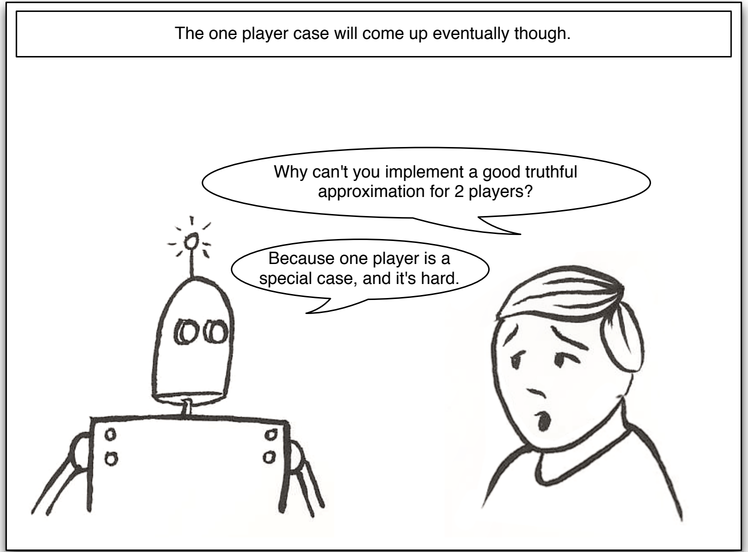
Communication Complexity



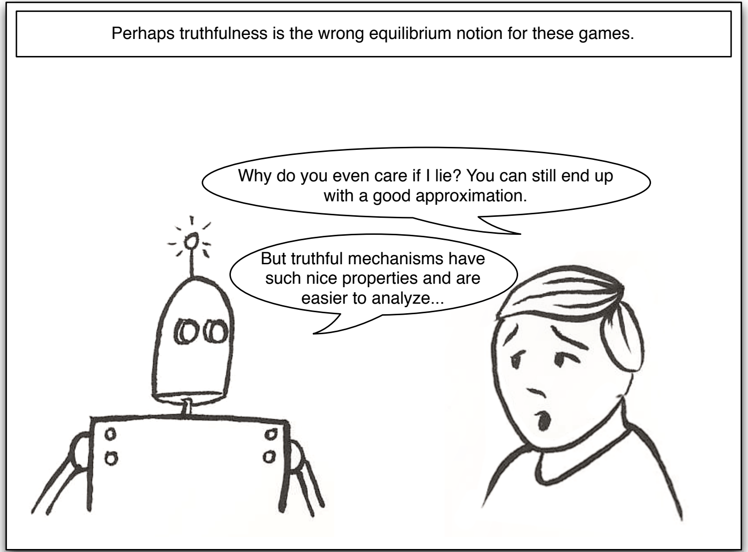
Is It Even a Game?



Is It Even a Game?



Maybe Truthfulness is Wrong



Summary of Issues

- Mechanisms are computationally limited, but players are not
- This asymmetry leads to hardness results that should not be
- Existing methods for resolving this asymmetry do not work

Outline

Instance Oracle Definition

We developed a model in which the mechanism can query information about valuation functions to solve an instance.

Definition (Instance Oracle)

An instance oracle answers queries related to specific problem instances. Let $a \in A$ be an instance of a problem A. We define an oracle O such that

- Queries to O are made in the form of a string x
- O returns some function $O(a, x)$

We denote the pairing of A with O by A^O.

Instance Oracles

Example (Demand Oracle)

A demand oracle takes in a set of per-item prices $p_1, \ldots, p_m$ and returns a set S maximizing

$$v(S) - \sum_{j \in S} p_j$$

We empower mechanisms with access to oracles.

Benefits of Our Model

- Efficient mechanisms in this model work well in practice if worst-case hardness is unnatural
- Hardness results do not depend on unnatural worst cases
- All single-player games are easy

Outline

Reductions

Suppose we have two allocation problems $\mathcal{A}$ and $\mathcal{B}$ and we want to show that $\mathcal{B}$ is at least as hard as $\mathcal{A}$.

- Usual answer: reduce $\mathcal{A}$ to $\mathcal{B}$
- Trickier: what if $\mathcal{A}$ and $\mathcal{B}$ are paired with oracles?

Reductions

Suppose we have two allocation problems $\mathcal{A}$ and $\mathcal{B}$ and we want to show that $\mathcal{B}$ is at least as hard as $\mathcal{A}$.

- Usual answer: reduce $\mathcal{A}$ to $\mathcal{B}$
- Trickier: what if $\mathcal{A}$ and $\mathcal{B}$ are paired with oracles?

If $\mathcal{A}^O$ reduces to $\mathcal{B}^Q$, we want

- A poly-time solution to $\mathcal{B}^Q$ implies one for $\mathcal{A}^O$
- No poly-time solution to $\mathcal{A}^O$ implies none for $\mathcal{B}^Q$
- If $\mathcal{B}^Q$ reduces to $\mathcal{C}^U$, so does $\mathcal{A}^O$

Reductions

Suppose we have two allocation problems $\mathcal{A}$ and $\mathcal{B}$ and we want to show that $\mathcal{B}$ is at least as hard as $\mathcal{A}$.

- Usual answer: reduce $\mathcal{A}$ to $\mathcal{B}$
- Trickier: what if $\mathcal{A}$ and $\mathcal{B}$ are paired with oracles?

If $\mathcal{A}^O$ reduces to $\mathcal{B}^Q$, we want

- A poly-time solution to $\mathcal{B}^Q$ implies one for $\mathcal{A}^O$
- If $\mathcal{B}^Q$ reduces to $\mathcal{C}^U$, so does $\mathcal{A}^O$

Reductions Definition

Reducing $\mathcal{A}^O$ to $\mathcal{B}^Q$

1. Find a polynomial-time reduction R from $\mathcal{A}$ to $\mathcal{B}$
2. Show that if $R(a) = b$, queries to Q on b can be answered in polynomial time with access to O on a

$$\mathcal{A} \xrightarrow[R]{} \mathcal{B}$$

$$\begin{tikzcd}
 & & O \arrow[dl, dash] & & \\
 & O(a,y) \arrow[dr] & & (a,y) \arrow[ul] \arrow[dl, dash] & \\
(b,x) \arrow[rr] & & f \arrow[rr] & & Q(b,x)
\end{tikzcd}$$

Making Reductions Work

If $\mathcal{A}^O$ reduces to $\mathcal{B}^Q$, we want

- A poly-time solution to $\mathcal{B}^Q$ implies one for $\mathcal{A}^O$
- If $\mathcal{B}^Q$ reduces to $\mathcal{C}^U$, so does $\mathcal{A}^O$

$$\mathcal{A} \xrightarrow[R]{} \mathcal{B}$$

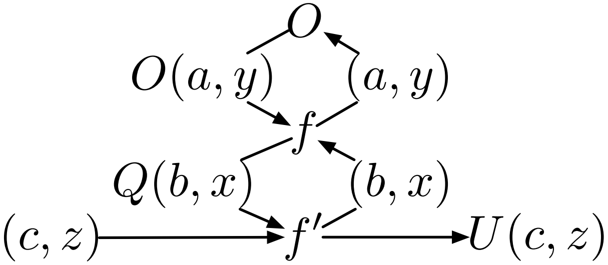
$$\begin{tikzcd}
 & & O & & \\
 & O(a,y) \arrow[ur, dash] \arrow[dr] & & (a,y) \arrow[ul] \arrow[dl, dash] & \\
(b,x) \arrow[rr] & & f \arrow[rr] & & Q(b,x)
\end{tikzcd}$$

Making Reductions Work

If $\mathcal{A}^O$ reduces to $\mathcal{B}^Q$, we want

- A poly-time solution to $\mathcal{B}^Q$ implies one for $\mathcal{A}^O$
- If $\mathcal{B}^Q$ reduces to $\mathcal{C}^U$, so does $\mathcal{A}^O$

$$\mathcal{A} \xrightarrow{R} \mathcal{B} \xrightarrow{R'} \mathcal{C}$$



Completeness

Reducing $\mathcal{A}^O$ to $\mathcal{B}^Q$

1. Find a polynomial-time reduction R from $\mathcal{A}$ to $\mathcal{B}$
2. Show that if $R(a) = b$, queries to Q on b can be answered in polynomial time with access to O on a

We define IONP to be the class of problems $\mathcal{A}^O$ with $\mathcal{A} \in \mathrm{NP}$.

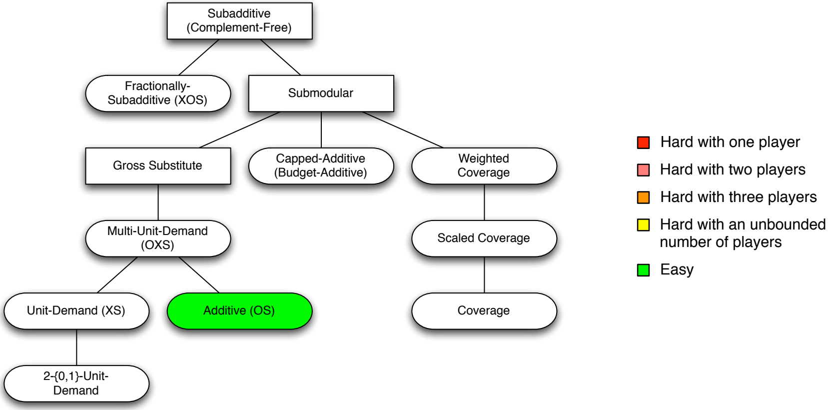
Any of the following conditions show that $\mathcal{A}^O$ is IONP-hard:

- $\mathcal{A}$ is NP-hard and queries to O are poly-time computable
- $\mathcal{B}^Q$ is IONP-hard and reduces to $\mathcal{A}^O$
- $\mathcal{A}$ is shown NP-hard via R and O is easy on $R(a)$

Outline

Simple Results

We begin by showing that oracle queries are easy for several of the classes of public projects we study.



Oracle Definitions

Definition (k-Demand Oracle)

A k-demand oracle takes in a list of prices $p_1, \ldots, p_m$ and returns a set S of size k maximizing $v_i(S) - \sum_{j \in S} p_j$

Definition (Demand Oracle)

A demand oracle takes in a list of prices $p_1, \ldots, p_m$ and returns a set S maximizing $v_i(S) - \sum_{j \in S} p_j$

How to compute k-demand queries

Theorem

Let $\mathcal{V}$ be a valuation class for which 2-player public projects have a polynomial-time exact solution. k-demand queries can be solved exactly in polynomial time for valuations in $\mathcal{V}$.

Proof.

Consider a query $p_1, \ldots, p_m$ to a valuation function v_i.

1. Let $P = \max_j p_j$
2. Let $v_i'(S) = \sum_{j \in S}(P - p_j)$
3. Solve the public project with players v_i, v_i' to get S

$\square$

How to compute k-demand queries

Theorem

Let $\mathcal{V}$ be a valuation class for which 2-player public projects have a polynomial-time exact solution. k-demand queries can be solved exactly in polynomial time for valuations in $\mathcal{V}$.

Proof.

Consider a query $p_1, \ldots, p_m$ to a valuation function v_i.

1. Let $P = \max_j p_j$
2. Let $v'_i(S) = \sum_{j \in S}(P - p_j)$
3. Solve the public project with players v_i, v'_i to get S

S maximizes $v_i(S) + v'_i(S) = v_i(S) - \sum_{j \in S} p_j + kP$, and is therefore an answer to the k-demand query. $\square$

How to compute demand queries

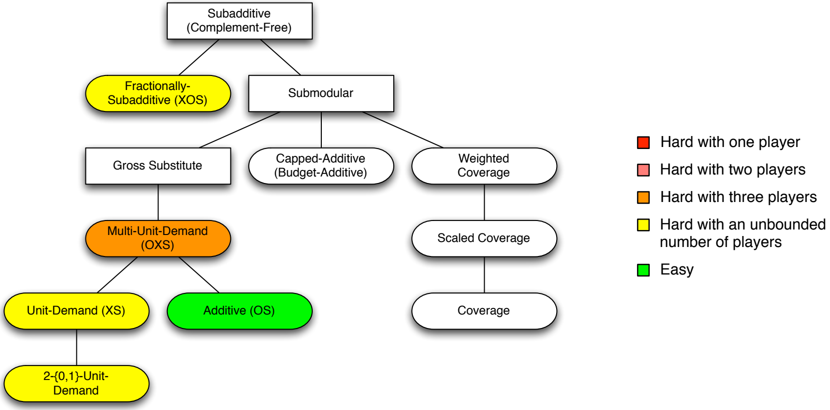
Theorem

Let $\mathcal{V}$ be a valuation class for which 2-player auctions have a polynomial-time exact solution. Demand queries can be solved exactly in polynomial time for valuations in $\mathcal{V}$.

This theorem has a similar proof to the one for k-demand queries.

Results following from easy oracles

All the classes of public projects which are easy for two players have easy oracles, so oracles do not affect their complexity.



Outline

What We Show

Theorem

Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.

Definition (Coverage Valuation)

A coverage valuation v_i consists of m sets $V_i^1, \ldots, V_i^m$ and the value of a set S is

$$v_i(S) = \left| \bigcup_{j \in S} V_i^j \right|$$

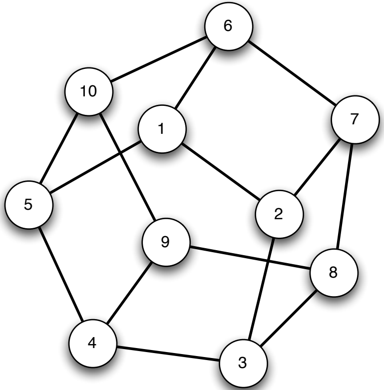
- These public projects are NP-hard with 1 player, so we can't show that oracle queries are easy
- We show a reduction to a special case where oracles are easy

The Reduction

Theorem

Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.

- We reduce from vertex cover on a 3-regular graph

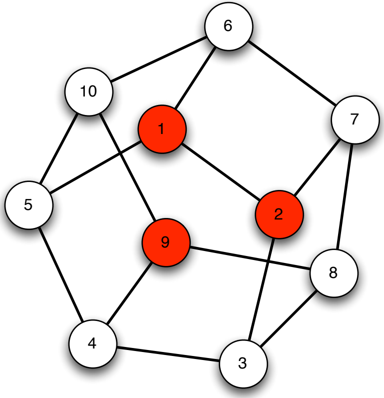


The Reduction

Theorem

Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.

- We reduce from vertex cover on a 3-regular graph

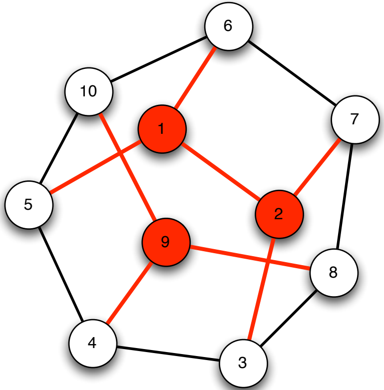


The Reduction

Theorem

Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.

- We reduce from vertex cover on a 3-regular graph

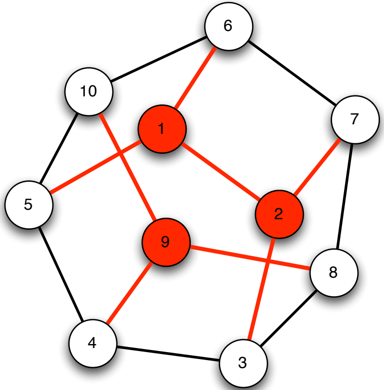


The Reduction

Theorem

Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.

- We reduce from vertex cover on a 3-regular graph
- We can construct $v(S) = \#$ edges covered by S

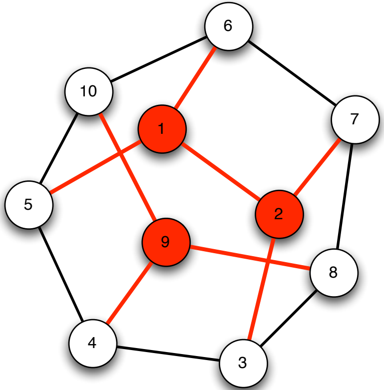


The Reduction

Theorem

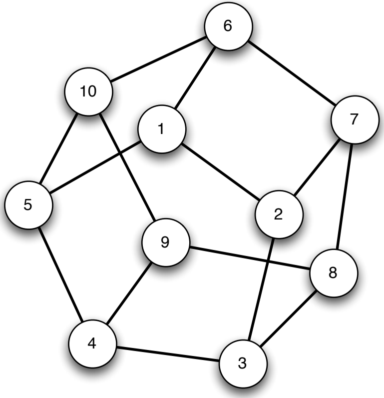
Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.

- We reduce from vertex cover on a 3-regular graph
- We can construct $v(S) = \#$ edges covered by S
- We split the graph into two simpler graphs where queries are easy
- Simultaneously maximizing both valuations is hard



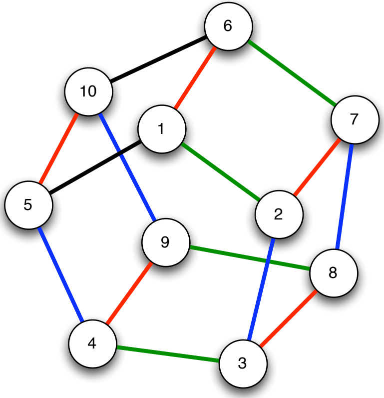
The Reduction

We begin with an instance of vertex cover on a 3-regular graph.



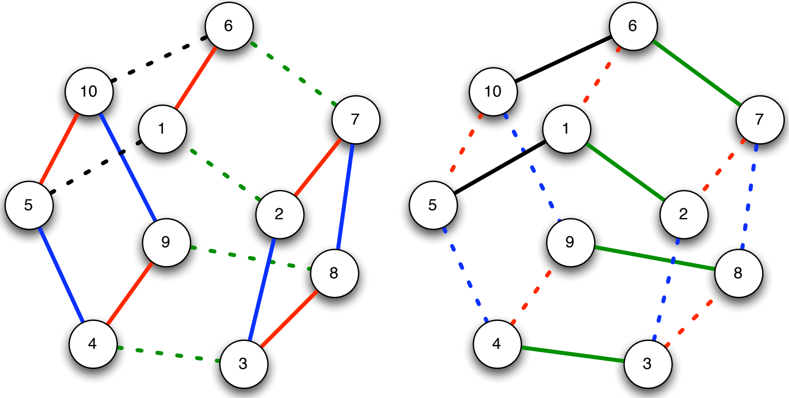
Four-Coloring

First, 4-color the edges (possible by Vizing's theorem)



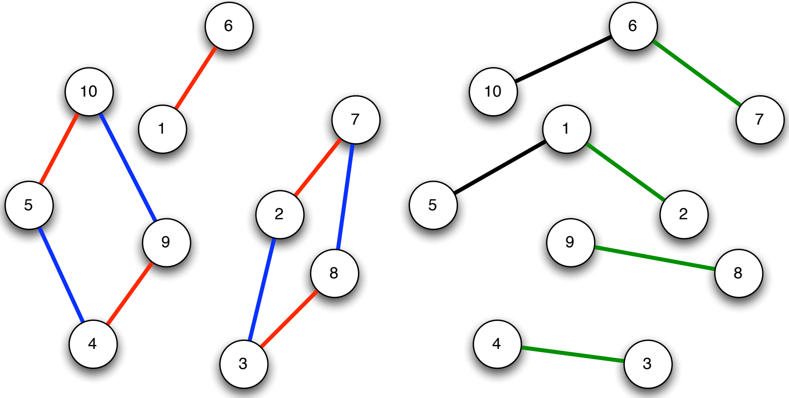
Split the Graph

Partition the edges by colors to get two 2-colorable graphs



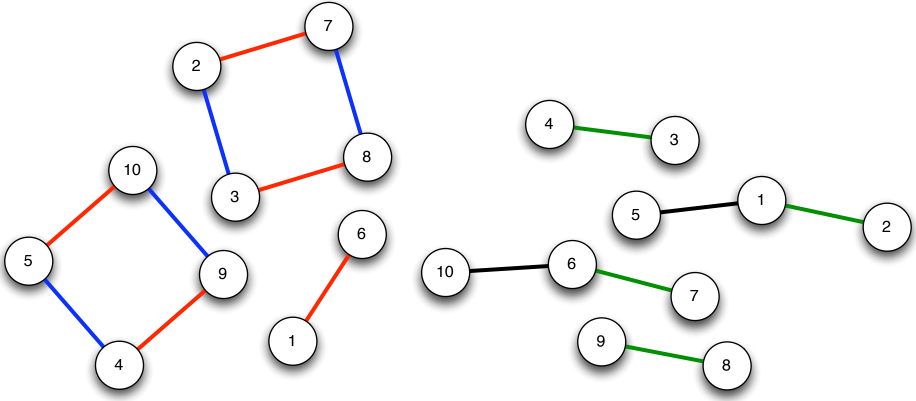
Split the Graph

Partition the edges by colors to get two 2-colorable graphs



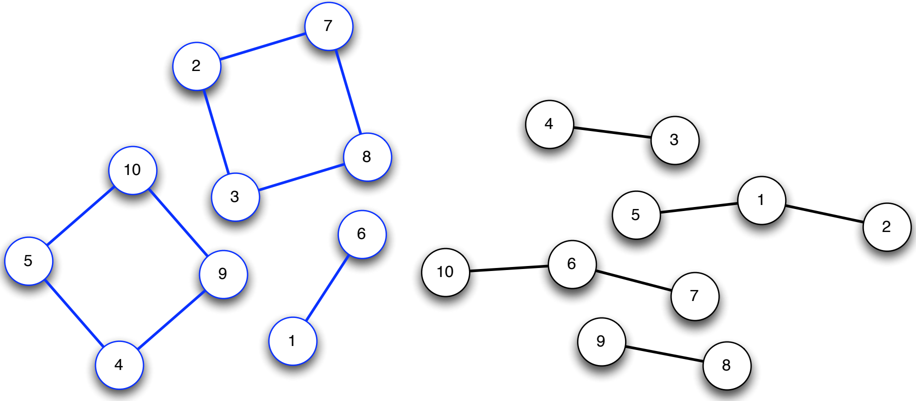
Split the Graph

Partition the edges by colors to get two 2-colorable graphs



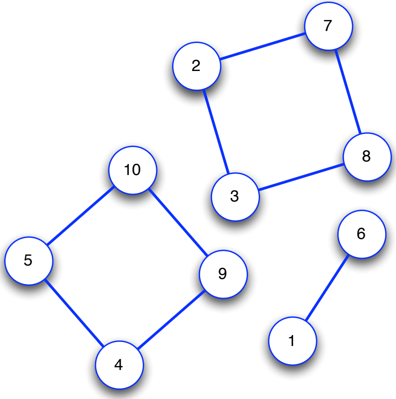
Split the Graph

Partition the edges by colors to get two 2-colorable graphs



A set of nodes is a vertex cover in the original graph iff it covers both of these graphs, so we have a valid reduction here.

Computing Queries



It's easy to compute queries on paths and cycles

Results

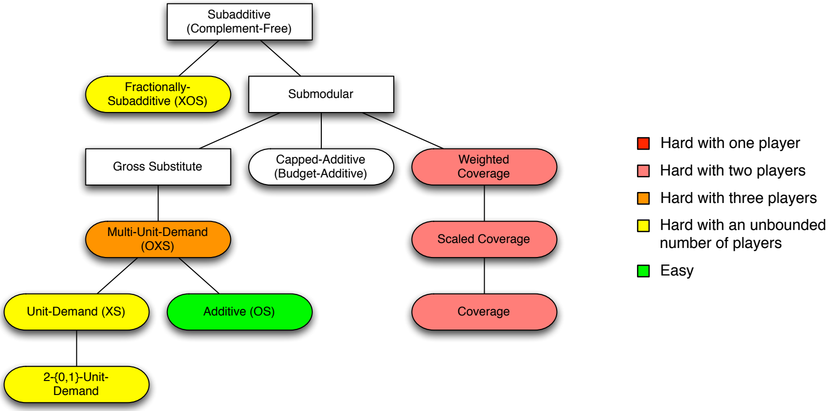
Theorem

Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.

Results

Theorem

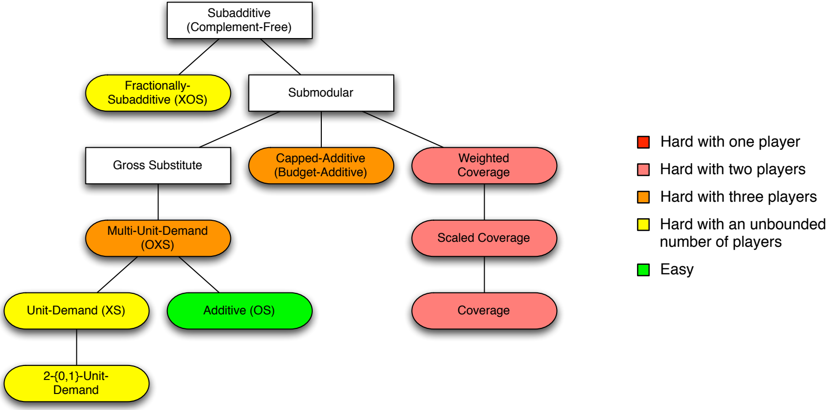
Public projects with 2 players with coverage valuations and k-demand or demand oracles are IONP hard.



Outline

Summary of Results

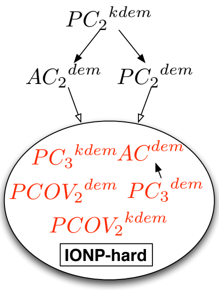
We were able to show hardness for all classes using oracles.



The only case we missed was 2 capped-additive players.

2 Capped-Additive Players

We showed that public projects with 2 capped-additive players and k-demand queries reduce to both public projects and auctions with 2 capped-additive players and demand queries.



P: public project
A: auction
C: capped-additive valuation
COV: coverage valuation
subscript: number of players

Outline

Tables of Results

Bounds on best achievable approximation ratio r for public projects

	Number of Players		
Valuation Class	1	2	3
Additive	1	1	1
Unit-Demand	1	1	1
Multi-Unit-Demand	1	1 [New]	1 [New] $< r \leq 3/2$ [New]
Capped-Additive	1	$1+\epsilon$ [New]	$1+\epsilon$ [New]
Coverage	$r = e/(e-1)$	$r = e/(e-1)$	$r = e/(e-1)$
Fractionally-Subadditive	1	1	1

	Number of Players	
Valuation Class	Constant	Unbounded
Additive	1	1
Unit-Demand	1	$r = e/(e-1)$ [New]
Multi-Unit-Demand	$1+\epsilon$ [New] $< r \leq e/(e-1)$ [New]	$r = e/(e-1)$ [New]
Capped-Additive	$r = 1+\epsilon$ [New]	$r = e/(e-1)$ [New]
Coverage	$r = e/(e-1)$	$r = e/(e-1)$
Fractionally-Subadditive	1	$r \geq 2^{\log^{1-\gamma}(\min(n,m))}$ [New]

Tables of Results

Best achievable MIR approximations for public projects

	Number of Players				
Valuation Class	1	2	3	Constant	Unbounded
Additive	1	1	1	1	1
Unit-Demand	1	1	1	1	$\sqrt{m}$ [New]
Multi-Unit-Demand	1	1 [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]
Capped-Additive	1	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]
Coverage	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]
Fractionally-Subadditive	1	1	1	1	$\sqrt{m}$ [New]

Best MIR approximations with demand or k-demand oracles

	Number of Players				
Valuation Class	1	2	3	Constant	Unbounded
Additive	1	1	1	1	1
Unit-Demand	1	1	1	1	$\sqrt{m}$ [New]
Multi-Unit-Demand	1	1 [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]
Capped-Additive	1	?	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]
Coverage	1/?	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]	$\sqrt{m}$ [New]
Fractionally-Subadditive	1	1	1	1	$\sqrt{m}$ [New]

Conclusions and Open Problems

Conclusions

- Truthful mechanisms for submodular valuations are hard
- Hardness is mostly preserved even with oracle access

Open Problems

- What is the complexity of ${\mathrm{PC}_2}^{kdem}$, ${\mathrm{PC}_2}^{dem}$, ${\mathrm{PCOV}_1}^{dem}$?
- Are there reasonable oracles that make some of our hard problems easy?
- Auctions remain largely open in our oracle framework
- Our framework is an interesting tool that can be used to study other problems

Acknowledgements

Thanks to everyone who helped me get to this point.

- Shaddin Dughmi
- John Ledyard
- Michael Schapira
- Leonard Schulman
- Yaron Singer
- Chris Umans
- Adam Wierman

In Theaters Now

3-Player Capped-Additive Public Projects

Definition (Capped-Additive)

A capped-additive valuation function v_i has values $v_i^1, \dots, v_i^m$ for items $1, \dots, m$ and a value cap c_i. The value for a set S is

$$v_i(S) = \min\left(\sum_{j \in S} v_i^j, c_i\right).$$

3-Dimensional Matching

Definition (3DM)

An instance of 3DM consists of a set $T \subseteq [k] \times [k] \times [k]$. Is there some $S \subseteq T$ of size $|S| = k$ such that $\forall i, j \exists (x_1, x_2, x_3) \in S$ such that $x_i = j$?

Equivalently, does there exist an S of size k such that the projection of S onto any of its coordinates is $[k]$?

Single Coordinate Reduction

Question

Does there exist an S of size k such that the projection of S onto any of its coordinates is $[k]$?

Consider a single coordinate. Let $T_i = \{x_i^1, \ldots, x_i^m\}$ be the projection of T to coordinate i. Does there exist some $S_i \subseteq T_i$ where $|S_i| = k$ and $S_i = [k]$?

Easy to answer, but let's try a reduction:

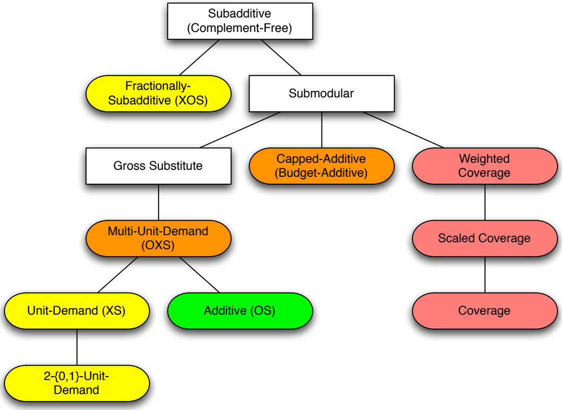
- Player 1 has $v_1^j = 2^{x_i^j}$ and $c_1 = 2^{m+1} - 1$
- Player 2 has $v_2^j = 2^{m+1} - 2^{x_i^j}$ and $c_2 = k2^{m+1} - (2^{m+1} - 1)$

$S_i = [k]$ iff $v_1(S) = c_1$ and $v_2(S) = c_2$.

3-Coordinate Reduction

- We saw that 2 players are enough to perform a reduction that checks a single coordinate. So 6 players are enough to do this for all 3 coordinates.
- Each player either has positive or negative value for items in their coordinate
- We could combine players such that a single player has values corresponding to multiple coordinates
- If we reduce to 2 players, each player has 3 coordinates, so queries must solve 3DM
- If we reduce to 3 players, each player has 2 coordinates, so we need only solve bipartite matching

Cumulative Results



Capped-Additive Auctions

So far, our reductions haven't made use of oracles. We now reduce from capped-additive public projects with demand oracles to capped-additive auctions with demand oracles.

The Reduction

Start with a public project with m items, of which we allocate k

$$1 \quad 2 \quad 3 \quad 4 \quad \cdots \quad m$$

The Reduction

Create n duplicates of the items, one for each player

$$
\begin{array}{cccccc}
1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1_n & 2_n & 3_n & 4_n & \cdots & m_n
\end{array}
$$

The Reduction

Create n duplicates of the items, one for each player

$$\begin{array}{ccccccc}
v_1 & \color{red}{1_1} & \color{red}{2_1} & \color{red}{3_1} & \color{red}{4_1} & \color{red}{\cdots} & \color{red}{m_1} \\
 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
 & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & 1_n & 2_n & 3_n & 4_n & \cdots & m_n
\end{array}$$

The Reduction

Create n duplicates of the items, one for each player

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & \color{red}{1_2} & \color{red}{2_2} & \color{red}{3_2} & \color{red}{4_2} & \color{red}{\cdots} & \color{red}{m_2} \\
 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
 & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & 1_n & 2_n & 3_n & 4_n & \cdots & m_n
\end{array}
$$

The Reduction

Create n duplicates of the items, one for each player

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & \color{red}{1_3} & \color{red}{2_3} & \color{red}{3_3} & \color{red}{4_3} & \color{red}{\cdots} & \color{red}{m_3} \\
 & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 & 1_n & 2_n & 3_n & 4_n & \cdots & m_n
\end{array}
$$

The Reduction

Create n duplicates of the items, one for each player

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & \color{red}{1_n} & \color{red}{2_n} & \color{red}{3_n} & \color{red}{4_n} & \color{red}{\cdots} & \color{red}{m_n}
\end{array}
$$

The Reduction

Create m additional players, one for each original item

$$\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & m_n
\end{array}$$

The Reduction

Create m additional players, one for each original item

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & m_n \\
 & v_{n+1} & & & & &
\end{array}
$$

The Reduction

Create m additional players, one for each original item

$$
\begin{array}{ccccccc}
v_1 & 1_1 & \color{red}{2_1} & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & \color{red}{2_2} & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & \color{red}{2_3} & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \color{red}{\vdots} & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & \color{red}{2_n} & 3_n & 4_n & \cdots & m_n \\
 & v_{n+1} & v_{n+2} & & & &
\end{array}
$$

The Reduction

Create m additional players, one for each original item

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & \color{red}{3_1} & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & \color{red}{3_2} & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & \color{red}{3_3} & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \color{red}{\vdots} & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & \color{red}{3_n} & 4_n & \cdots & m_n \\
 & v_{n+1} & v_{n+2} & v_{n+3} & & &
\end{array}
$$

The Reduction

Create m additional players, one for each original item

$$\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & \color{red}{4_1} & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & \color{red}{4_2} & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & \color{red}{4_3} & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \color{red}{\vdots} & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & \color{red}{4_n} & \cdots & m_n \\
 & v_{n+1} & v_{n+2} & v_{n+3} & v_{n+4} & &
\end{array}$$

The Reduction

Create m additional players, one for each original item

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & \color{red}{m_1} \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & \color{red}{m_2} \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & \color{red}{m_3} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \color{red}{\vdots} \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & \color{red}{m_n} \\
 & v_{n+1} & v_{n+2} & v_{n+3} & v_{n+4} & \cdots & v_{n+m}
\end{array}
$$

The Reduction

Create k additional items which each player v_{n+j} values at c_{n+j}

$$\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & m_n \\
 & v_{n+1} & v_{n+2} & v_{n+3} & v_{n+4} & \cdots & v_{n+m}
\end{array}$$

The Reduction

Create k additional items which each player v_{n+j} values at c_{n+j}

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & m_n \\
 & {\color{red}v_{n+1}} & v_{n+2} & {\color{red}v_{n+3}} & v_{n+4} & \cdots & v_{n+m}
\end{array}
$$

- k players are satisfied by these k items

The Reduction

Create k additional items which each player v_{n+j} values at c_{n+j}

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & m_n \\
 & v_{n+1} & v_{n+2} & v_{n+3} & v_{n+4} & \cdots & v_{n+m}
\end{array}
$$

- k players are satisfied by these k items
- The other $m - k$ need their whole column

The Reduction

Create k additional items which each player v_{n+j} values at c_{n+j}

$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & m_n \\
 & v_{n+1} & v_{n+2} & v_{n+3} & v_{n+4} & \cdots & v_{n+m}
\end{array}
$$

- k players are satisfied by these k items
- The other $m-k$ need their whole column
- The original n players get the same k items each

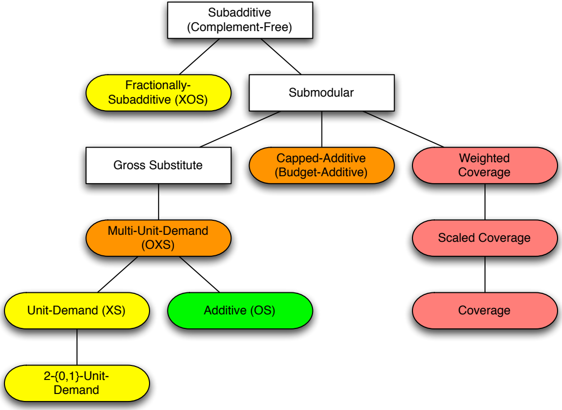
The Reduction

Create k additional items which each player v_{n+j} values at c_{n+j}

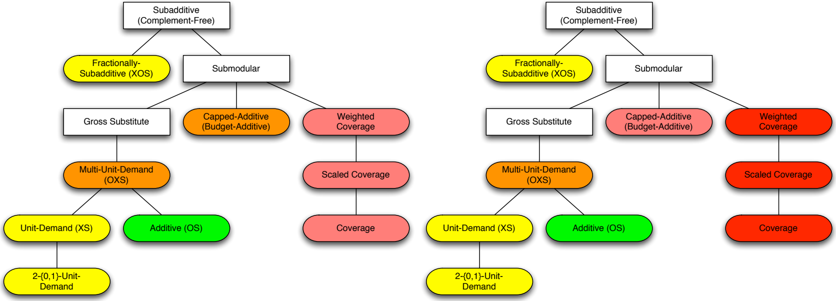
$$
\begin{array}{ccccccc}
v_1 & 1_1 & 2_1 & 3_1 & 4_1 & \cdots & m_1 \\
v_2 & 1_2 & 2_2 & 3_2 & 4_2 & \cdots & m_2 \\
v_3 & 1_3 & 2_3 & 3_3 & 4_3 & \cdots & m_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
v_n & 1_n & 2_n & 3_n & 4_n & \cdots & m_n \\
 & v_{n+1} & v_{n+2} & v_{n+3} & v_{n+4} & \cdots & v_{n+m}
\end{array}
$$

- k players are satisfied by these k items
- The other $m-k$ need their whole column
- The original n players get the same k items each
- Demand queries can all be answered

Reductions Without Completeness



Reductions Without Completeness



This picture leaves open 2 capped-additive players.

Two Kinds of Oracles

We denote the 2-player capped-additive public project with k-demand and demand oracles by ${\mathrm{PC}_2}^{kdem}$ and ${\mathrm{PC}_2}^{dem}$.

We don't yet know whether ${\mathrm{PC}_2}^{kdem}$ and ${\mathrm{PC}_2}^{dem}$ are IONP-hard, but we can reduce between them.

k-Demand to Demand Reduction

Consider a public project where player i has valuation function

$$v_i(S) = \min(\sum_{j \in S} v_i^j, c_i).$$

Let $V = \sum_{i,j} v_i^j$ and

$$v_i'(S) = \min\left(\sum_{j \in S}\left(v_i^j + V\right), c_i + kV\right)$$

The social welfare of a set S of size k after this reduction is just the social welfare before it, plus nkV. So the welfare-maximizing set is not changed.

Oracle Queries

$$v_i'(S) = \min\left(\sum_{j \in S}\left(v_i^j + V\right), c_i + kV\right)$$

- If the demand query returns a set of size k, the k-demand query can tell us which set it is
- If the demand query returns a set of size $< k$, the cap isn't reached, so it's as easy as additive
- If the demand query returns a set of size $> k$, the cap is reached, so we only need to minimize the price

Reduction to Auctions

We can reduce not only across oracles, but from public projects to auctions as well. Let ${\mathrm{AC}_2}^{\mathit{dem}}$ be the 2-player combinatorial auction problem with capped-additive valuations and demand queries.

The Reduction

Consider an $\mathrm{AC}_2{}^{dem}$ instance with values

$$v_i(S) = \min\left(\sum_{j\in S} v_i^j, c_i\right).$$

Let $V = \sum_{i,j} v_i^j$ and $W = \sum_j v_2^j$. We produce an instance with valuations

$$\begin{aligned} v_1'(S) &= \min\left(\sum_{j\in S}\left(v_1^j + V\right), kV + c_1\right) \\ v_2'(S) &= \min\left(\sum_{j\in S}\left(V - v_2^j\right), (m-k)V - (W - c_2)\right) \end{aligned}$$

Reduction is Valid

$$v'_1(S) = \min\left(\sum_{j\in S}\left(V + v_1^j\right), kV + c_1\right)$$

$$v'_2(S) = \min\left(\sum_{j\in S}\left(V - v_2^j\right), (m-k)V - (W - c_2)\right)$$

- In an optimal allocation, player 1 gets a set S of size k and player 2 gets S^C
- The social welfare of such an allocation is equal to the social welfare of the $\mathrm{PC_2}^{kdem}$ instance, plus some fixed terms

Computing Oracle Queries

$$
\begin{aligned}
v_1'(S) &= \min\left(\sum_{j\in S}\left(V + v_1^j\right), kV + c_1\right) \\
v_2'(S) &= \min\left(\sum_{j\in S}\left(V - v_2^j\right), (m-k)V - (W - c_2)\right)
\end{aligned}
$$

- If player 1 doesn't get k items, queries are easy
- If player 1 gets k items, the k-demand oracle gives the answer
- If player 2 doesn't get $m-k$ items, queries are easy
- If player 2 gets $m-k$ items, the k-demand oracle can give us a set S of size k which is the complement of the best $m-k$

Bibliography I

- Dave Buchfuhrer, Shaddin Dughmi, Hu Fu, Robert Kleinberg, Elchanan Mossel, Christos H. Papadimitriou, Michael Schapira, Yaron Singer, and Christopher Umans.
  Inapproximability for VCG-based combinatorial auctions.
  Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 518–536, 2010.

- Dave Buchfuhrer, Michael Schapira, and Yaron Singer.
  Computation and incentives in combinatorial public projects.
  Proceedings of the 11th ACM Conference on Electronic Commerce (EC), pages 33–42, 2010.

Bibliography II

Shahar Dobzinski, Noam Nisan, and Michael Schapira.
Approximation algorithms for combinatorial auctions with complement-free bidders.
In *STOC*, 2005.

Noam Nisan and Amir Ronen.
Computationally feasible VCG mechanisms.
Journal of Artificial Intelligence Research (JAIR), 29:19–47, 2007.

Michael Schapira and Yaron Singer.
Inapproximability of combinatorial public projects.
In Christos H. Papadimitriou and Shuzhong Zhang, editors, *WINE*, volume 5385 of *Lecture Notes in Computer Science*, pages 351–361. Springer, 2008.