CS599: Algorithm Design in Strategic Settings
Fall 2012
Lecture 7: Prior-Free Multi-Parameter Mechanism Design

Instructor: Shaddin Dughmi

Outline

Outline

Recall: Single Parameter Problems

Single-parameter Problem

- A homogenous good or service is being allocated (possibly under constraints).
- Player utilities linear in amount of good/service received.
- Player's private type is his value per unit good/service.

The special structure of these problems enables a simple characterization of dominant-strategy truthful mechanisms.

Myerson's Lemma

- An allocation rule (i.e. algorithm) for a single-parameter problem is implementable in dominant-strategies if and only if it is monotone.
- Moroever, the truth-telling payment rule is unique up to a bid-independent pivot term for each player.

Single-Parameter Problems are Permissive

Monotonicity is not too difficult to satisfy

- For most natural objectives that depend only on the allocation rule, such as welfare and various notions of “fairness” (e.g. makespan), the optimal algorithm is monotone.
- In most cases where the optimal algorithm is monotone, researchers were able to match the best polynomial-time approximation algorithm’s ratio by a monotone algorithm.
  - Related machine scheduling
  - Single-minded combinatorial auctions
  - Knapsack allocation
  - . . .

Single-Parameter Problems are Permissive

Monotonicity is not too difficult to satisfy

- For most natural objectives that depend only on the allocation rule, such as welfare and various notions of "fairness" (e.g. makespan), the optimal algorithm is monotone.
- In most cases where the optimal algorithm is monotone, researchers were able to match the best polynomial-time approximation algorithm's ratio by a monotone algorithm.
  - Related machine scheduling
  - Single-minded combinatorial auctions
  - Knapsack allocation
  - ...

Caveat

Some objectives are incompatible with truthfulness, polytime or not.

- E.g. single-item allocation with goal of minimizing the winning player's value.

Open Research Question

Consider an NP-hard single-parameter problem with an objective that depends only on the allocation rule. If there is truthful mechanism with approximation ratio α, and a polynomial-time algorithm with approximation ratio α, must there be a truthful polynomial-time mechanism with approximation ratio α?

Open Research Question

Consider an NP-hard single-parameter problem with an objective that depends only on the allocation rule. If there is truthful mechanism with approximation ratio α, and a polynomial-time algorithm with approximation ratio α, must there be a truthful polynomial-time mechanism with approximation ratio α?

In other words

For single-parameter problems, is truthfulness in polynomial time any "harder" than either truthfulness or polynomial time alone?

Open Research Question

Consider an NP-hard single-parameter problem with an objective that depends only on the allocation rule. If there is truthful mechanism with approximation ratio α, and a polynomial-time algorithm with approximation ratio α, must there be a truthful polynomial-time mechanism with approximation ratio α?

In other words

For single-parameter problems, is truthfulness in polynomial time any "harder" than either truthfulness or polynomial time alone?

So far as current research shows, the answer is conceivably yes, though some work has ruled out the most viable approaches to a positive answer (a black box reduction).

Beyond Single-parameter Problems

Empirically, there appears to be a "phase transition" in the difficulty of designing truthful mechanisms as we go beyond single-parameter problems.

- Little success, and impossibility results, for the design of approximate mechanisms (polynomial-time or not) for non-welfare-maximization problems.
- For problems where player typespaces allow the expression of too many valuations on Ω, characterizations that essentially limit truthful mechanisms to exact welfare maximization over a subset of Ω.

Beyond Single-parameter Problems

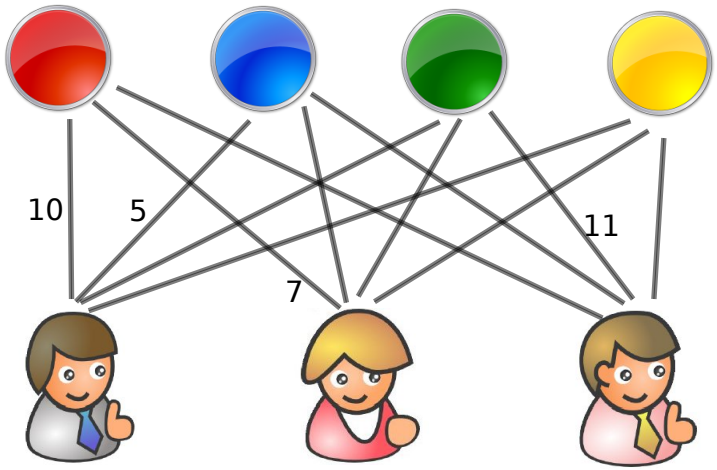
Empirically, there appears to be a "phase transition" in the difficulty of designing truthful mechanisms as we go beyond single-parameter problems.

- Little success, and impossibility results, for the design of approximate mechanisms (polynomial-time or not) for non-welfare-maximization problems.
- For problems where player typespaces allow the expression of too many valuations on Ω, characterizations that essentially limit truthful mechanisms to exact welfare maximization over a subset of Ω.

Next Up

Examples of Multi-parameter problems, the welfare-maximizing VCG mechanism, maximal-in-range algorithms, and characeterizations of dominant-strategy truthfulness.

Example: Matching

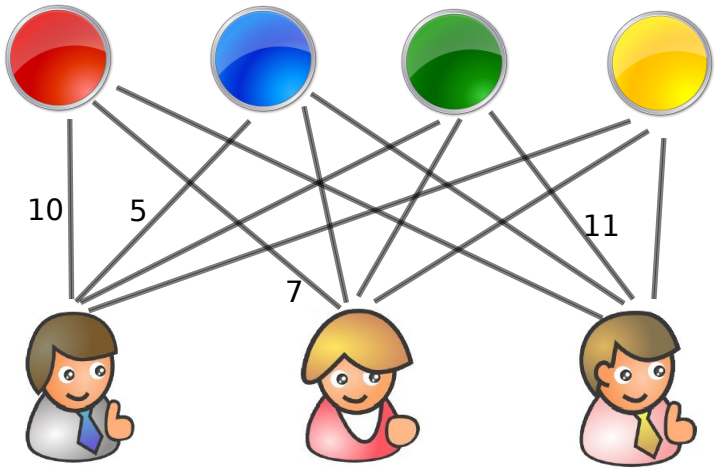


- n self-interested agents (the players), m items.
- Each player may receive at most one item.
- $v_i(j)$ is player i's value for item j (private)

Goal

Matching of items to players, at most one per player, maximizing total value of players (welfare).

Example: Matching



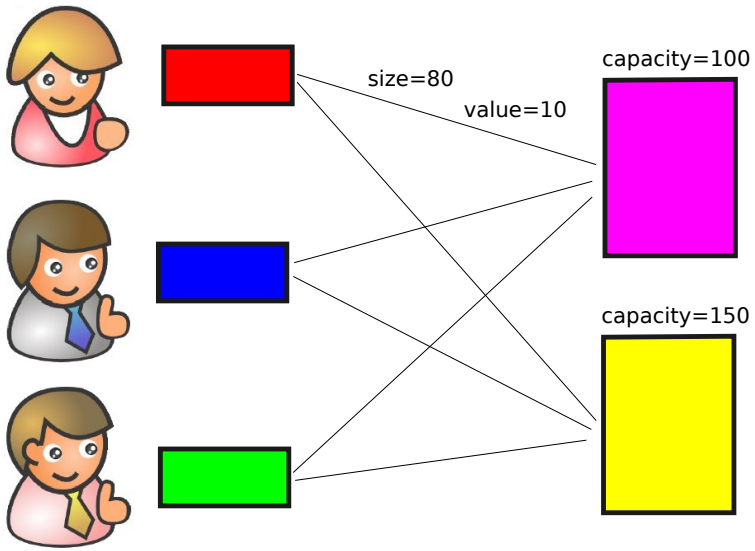
- n self-interested agents (the players), m items.
- Each player may receive at most one item.
- $v_i(j)$ is player i's value for item j (private)

Goal

Matching of items to players, at most one per player, maximizing total value of players (welfare).

Note: Generalization of adwords problem from HW1.

Example: Generalized Assignment

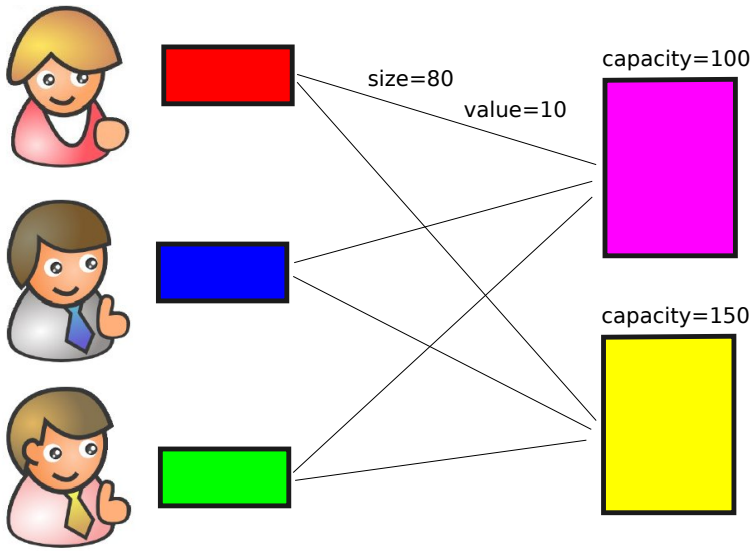


- n self-interested agents (the players), m machines.
- $s_i(j)$ is the size of player i's task on machine j. (public)
- C_j is machine j's capacity. (public)
- $v_i(j)$ is player i's value for his task going on machine j. (private)

Goal

Partial assignment of jobs to machines, respecting machine budgets, and maximizing total value of agents (welfare).

Example: Generalized Assignment



- n self-interested agents (the players), m machines.
- $s_i(j)$ is the size of player i's task on machine j. (public)
- C_j is machine j's capacity. (public)
- $v_i(j)$ is player i's value for his task going on machine j. (private)

Goal

Partial assignment of jobs to machines, respecting machine budgets, and maximizing total value of agents (welfare).

Note: When single machine, this is knapsack allocation.

Example: Unrelated Machine Scheduling

- n self-interested machines (the players), m tasks
- $t_i(j)$ is the time machine i takes to process task j. (private)

Goal

Schedule tasks on machines, with the goal of minimizing the completion time of all tasks (makespan).

Example: Unrelated Machine Scheduling

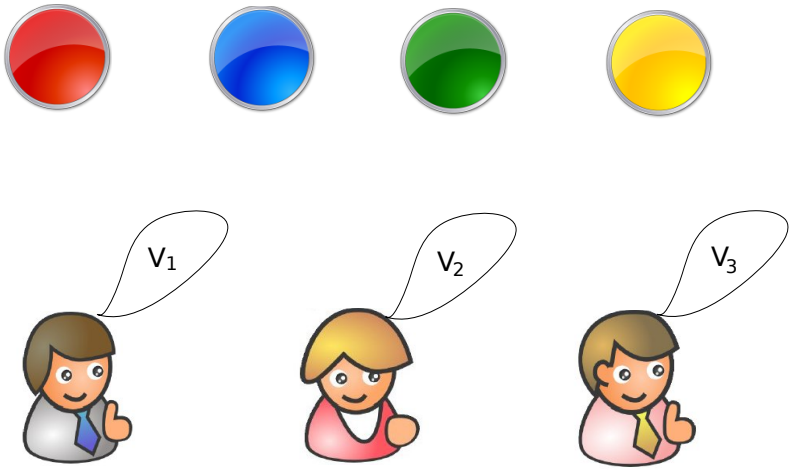
- n self-interested machines (the players), m tasks
- $t_i(j)$ is the time machine i takes to process task j. (private)

Goal

Schedule tasks on machines, with the goal of minimizing the completion time of all tasks (makespan).

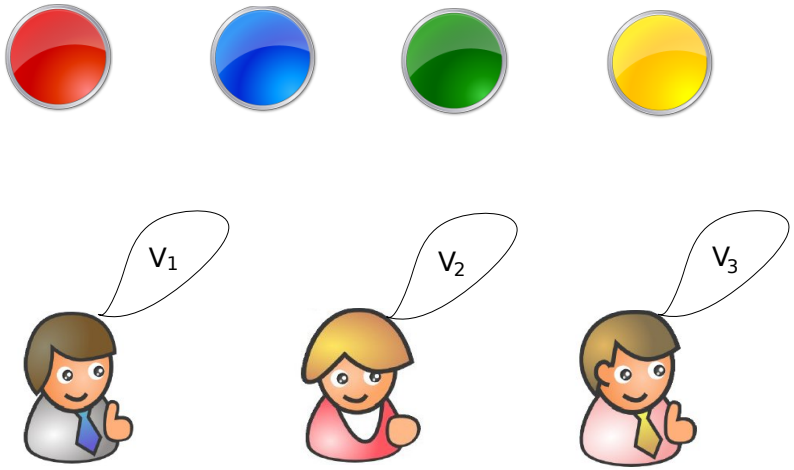
Note: When $t_i(j)/t_i(j') = t_{i'}(j)/t_{i'}(j')$ for all machines i,i', and tasks j,j', this is related machine scheduling which we studied last lecture.

Example: Combinatorial Allocation



- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

Example: Combinatorial Allocation

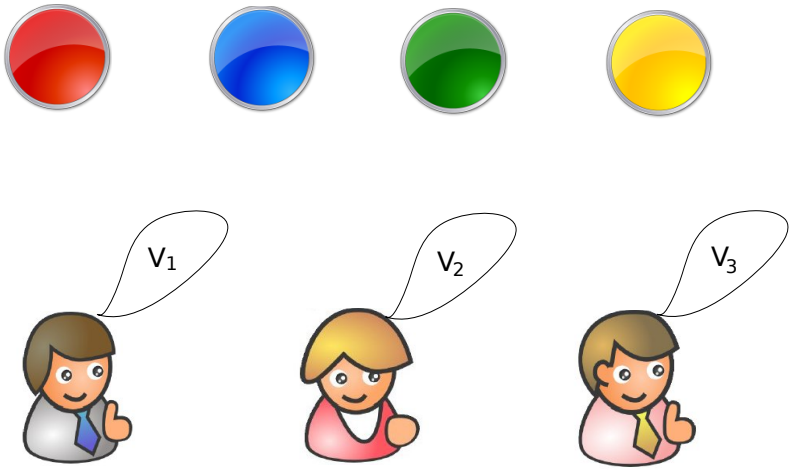


- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

Goal

Partition items into sets $S_1, S_2, \ldots, S_n$ to maximize welfare:
$v_1(S_1) + v_2(S_2) + \ldots v_n(S_n)$

Example: Combinatorial Allocation



- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

Goal

Partition items into sets $S_1, S_2, \ldots, S_n$ to maximize welfare: $v_1(S_1) + v_2(S_2) + \ldots v_n(S_n)$

Note: This is underspecified. We will in restrict valuations and assume a succinct representation.

Mechanism Design Problem in Quasi-linear Settings

Recall: Mechanism Design Problem

Public (common knowledge) inputs describes

- Set Ω of allocations.
- Typespace T_i for each player i.
  - $T = T_1 \times T_2 \times \ldots \times T_n$
- Valuation map $v_i : T_i \times \Omega \to \mathbb{R}$

Mechanism Design Problem in Quasi-linear Settings

Recall: Mechanism Design Problem

Public (common knowledge) inputs describes

- Set Ω of allocations.
- Typespace T_i for each player i.
  - $T = T_1 \times T_2 \times \ldots \times T_n$
- Valuation map $v_i : T_i \times \Omega \to \mathbb{R}$

Terminology Note

- When convenient, we think of the typespace T_i directly as a set functions mapping outcomes to the real numbers — i.e. $T_i \subseteq \mathbb{R}^\Omega$.
- In that case, we prefer denoting the typespace of player i by $\mathcal{V}_i \subseteq \mathbb{R}^\Omega$. Analogously, the set of valuation profiles is $\mathcal{V} = \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$.
- We refer to $\mathcal{V}_i$ also as the "valuation space" of player i, and each $v_i \in \mathcal{V}_i$ as a "private valuation" of player i.

Recall: Mechanisms and Truthfulness

Recall: Mechanism

A protocol of the following form, described by allocation rule $f : \mathcal{V} \to \Omega$, and payment rule $p : \mathcal{V} \to \mathbb{R}^n$, mapping private data to an allocation and payment for each player.

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Allocate according to $f(v_1, \ldots, v_n)$
3. Charge each player i payment $p_i(v_1, \ldots, v_n)$

Recall: Mechanisms and Truthfulness

Recall: Mechanism

A protocol of the following form, described by allocation rule $f : \mathcal{V} \to \Omega$, and payment rule $p : \mathcal{V} \to \mathbb{R}^n$, mapping private data to an allocation and payment for each player.

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Allocate according to $f(v_1, \ldots, v_n)$
3. Charge each player i payment $p_i(v_1, \ldots, v_n)$

Incentive-compatibility (Dominant Strategy)

A mechanism (f, p) is dominant-strategy truthful if, for every player i, valuation v_i, possible mis-report $\widehat{v}_i$, and reported valuations v_{-i} of the others, we have

$$\mathbf{E}[v_i(f(\vec{v})) - p_i(\vec{v})] \geq \mathbf{E}[v_i(f(\widehat{v}_i, v_{-i})) - p_i(\widehat{v}_i, v_{-i})]$$

The expectation is over the randomness in the mechanism.

Belated Note on Public vs Private Inputs

- In problems we consider, each legal input has a public portion (e.g. sizes of jobs in GAP), and a private portion (e.g. values of jobs in GAP).
- Public portion defines Ω, T_i, and $v_i : T_i \times \Omega \to \mathbb{R}^n$; i.e. defines the mechanism design setting.
- Technically, every "mechanism" we defined was a bunch of mechanisms, one for each legal choice of public data.
- However, as is traditional, we loosly refer to the entire algorithm that reads public and private data, and computes allocation and payments, as the "mechanism."
  - When we say such a "mechanism" is truthful, we mean the mechanism induced for each choice of public data is truthful.
  - When we say such a "mechanism" runs in polynomial time, we mean the algorithm that computes the allocation and payments from both the public and private data runs in polynomial time.

Design Goals

For each of the problems we described, we want a mechanism (allocation rule and payment rule) satisfying the following properties:

1. Dominant strategy Truthfulness
2. Individual rationality: payment from [to] player should be less than [greater than] his reported value [cost] for the allocation.
3. Polynomial time: The allocation algorithm must run in time polynomial in the number of bits used to describe the input.
4. Worst-case approximation ratio: As small as possible, given computational complexity assumptions.

Outline

Vickrey Clarke Groves (VCG) Mechanism

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Choose allocation $\omega^* \in \operatorname{argmax}_{\omega \in \Omega} \sum_i v_i(\omega)$
3. Charge each player i payment $h_i(v_{-i}) - \sum_{j \neq i} v_j(\omega^*)$

- Allocation rule maximizes welfare exactly over Ω
- Player i is paid the reported value of others for the chosen allocation, less a pivot term $h_i(v_{-i})$ independent of his own bid.

VCG is Truthful

Theorem

VCG is dominant-strategy truthful.

Proof

- Fix reports v_{-i} of players other than i.
- Assume player i's true valuation is v_i
- Player i's utility when reporting $\widehat{v}_i$ is given by

$$u_i(\widehat{v}_i) = v_i(\omega^*) + \sum_{j \neq i} v_j(\omega^*) - h_i(v_{-i}),$$

  where $\omega^* \in \operatorname{argmax}_{\omega \in \Omega} \left(\widehat{v}_i(\omega) + \sum_{j \neq i} v_j(\omega) \right)$

- Since the pivot term is independent of player i's bid, maximizing $u_i(\widehat{v}_i)$ is equivalent to maximizing

$$v_i(\omega^*) + \sum_{j \neq i} v_j(\omega^*)$$

- Setting $\widehat{v}_i = v_i$ then maximizes the above expression.
  - Interpretation: allow the mechanism to optimize player i's utility on his behalf

The Clarke Pivot Rule

For problems with non-negative valuations, there is a canonical choice for the pivot term that enforces individual rationality and non-negative transfers.

Clarke Pivot Rule

$h_i(v_{-i}) = \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega)$

Interpretation

- In VCG with the Clarke Pivot Rule, each player i pays the difference between $h_i(v_i)$ — the maximum welfare of players other than i — and the realized welfare of other players.
- In other words, player i pays the externality he imposes on others through participating in the mechanism.

Individual Rationality

Fact

Assume $\mathcal{V}_i \subseteq \mathbb{R}_+^\Omega$ for all players i. VCG with the Clarke Pivot Rule is individually rational — i.e. a truth-telling player's utility is always non-negative.

Individual Rationality

Proof

Utility of player when reporting his true valuation v_i, and others report v_{-i}, is

$$u_i(v) = v_i(\omega^*) + \sum_{j \neq i} v_j(\omega^*) - \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega)$$

$$= \sum_{j=1}^{n} v_j(\omega^*) - \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega)$$

Since the mechanism choses ω^* to maximize reported welfare, we have

$$u_i(v) = \max_{\omega \in \Omega} \sum_{j=1}^{n} v_j(\omega) - \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega)$$

By non-negativity of $v_i(\omega)$ for each $\omega \in \Omega$, this is non-negative.

Non-negative Transfers

Fact

VCG with the Clarke pivot rule does not pay players.

Non-negative Transfers

Fact

VCG with the Clarke pivot rule does not pay players.

Proof

Payment of player i is, by definition

$$p_i(v) = \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega) - \sum_{j \neq i} v_j(\omega^*)$$

This is clearly non-negative.

Applying VCG

Good News

In a sense, VCG is the best a utilitarian mechanism designer with unlimited computational power could hope for.

- Optimal for the welfare objective.
- Applies generally to any mechanism design problem (absent additional constraints on payments, e.g. budgets)
- If the algorithmic problem of finding a welfare maximizing allocation is polynomial-time solvable, then VCG can be implemented in polynomial time.
  - $n + 1$ calls to the algorithm, one for computing the allocation, and one per player to compute the Clarke pivot.

Applications: matching, routing, and many more.

Applying VCG

Bad News

- Specific to the welfare objective
  - As we will see later, this is unavoidable at this level of generality.
- Requires an exact algorithm for finding a welfare maximizing allocation, which is NP-hard for many problems.

Applying VCG

Bad News

- Specific to the welfare objective
  - As we will see later, this is unavoidable at this level of generality.
- Requires an exact algorithm for finding a welfare maximizing allocation, which is NP-hard for many problems.

Applying VCG

Bad News

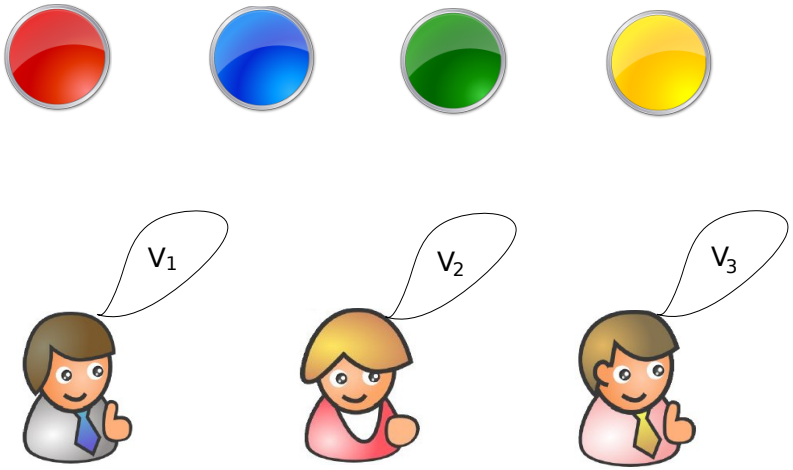
- Specific to the welfare objective
  - As we will see later, this is unavoidable at this level of generality.
- Requires an exact algorithm for finding a welfare maximizing allocation, which is NP-hard for many problems.

Next Up

A modification of the VCG mechanism that preserves truthfulness, relaxes exact optimization, and therefore sometimes recovers polynomial time implementability. Will illustrate through combinatorial allocation.

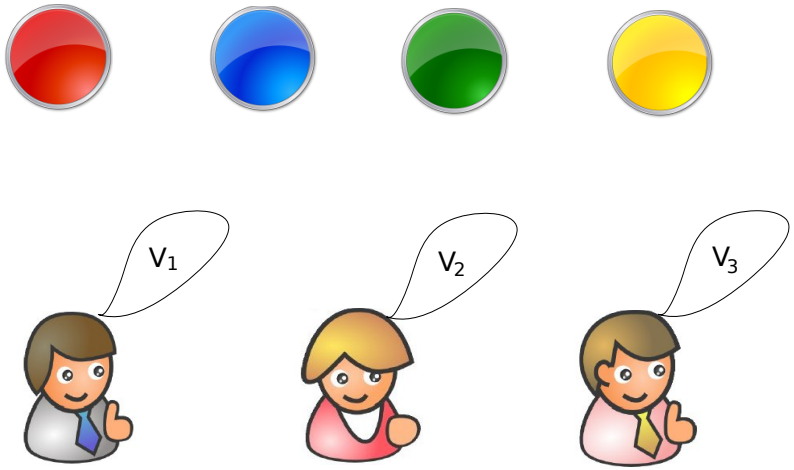
Outline

Recall: Combinatorial Allocation



- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

Recall: Combinatorial Allocation

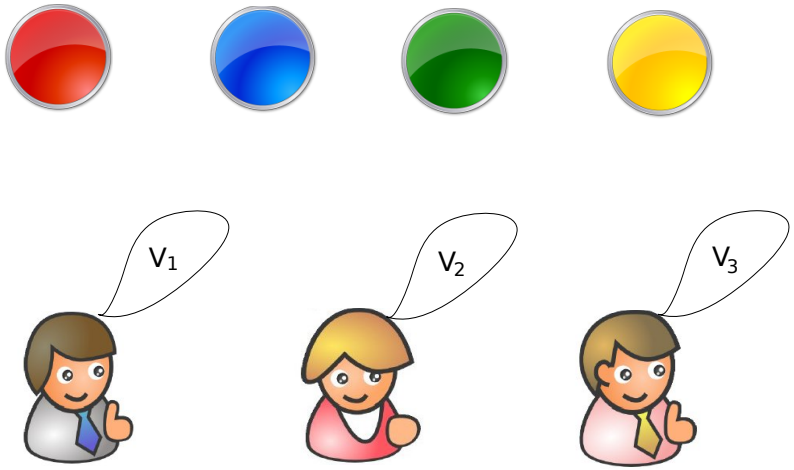


- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

Goal

Partition items into sets $S_1, S_2, \ldots, S_n$ to maximize welfare:
$v_1(S_1) + v_2(S_2) + \ldots v_n(S_n)$

Recall: Combinatorial Allocation



- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

Goal

Partition items into sets $S_1, S_2, \ldots, S_n$ to maximize welfare: $v_1(S_1) + v_2(S_2) + \ldots v_n(S_n)$

Note: This is underspecified. We will in restrict valuations and assume a succinct representation.

Specifying Typespaces in Combinatorial Allocation

Combinatorial Allocation (aka combinatorial auctions) is a family of problems, rather than one problem. A variant of CA is described by two things:

1. A class of valuations $\mathcal{V}_i : 2^{[m]} \to \mathbb{R}$; Typically, better positive results possible for restricted classes.
   - Note overload of notation.
2. One of the following:
   - A choice of representation, or language, to describe valuations in input, or more minimally
   - An oracle model, specifying each valuation $v_i \in \mathcal{V}_i$ is presented as black boxes that can answer certain questions about v_i. This often serves to quantify over many representations.

To introduce and illustrate the maximal-in-range technique, we will show a truthful $\sqrt{m}$ approximation mechanism for combinatorial allocation with subadditive valuations, in the value oracle model. The mechanism will run in time $\mathrm{poly}(n, m)$.

- Subadditivity: $v_i(S \bigcup T) \leq v_i(S) + v_i(T)$ for all $S, T \subseteq [m]$
- Value Oracle: v_i presented as a black-box which returns $v_i(S)$ on input S. Or, less generally, v_i is represented in some language such that $v_i(S)$ can be computed in $\mathrm{poly}(m)$ time.

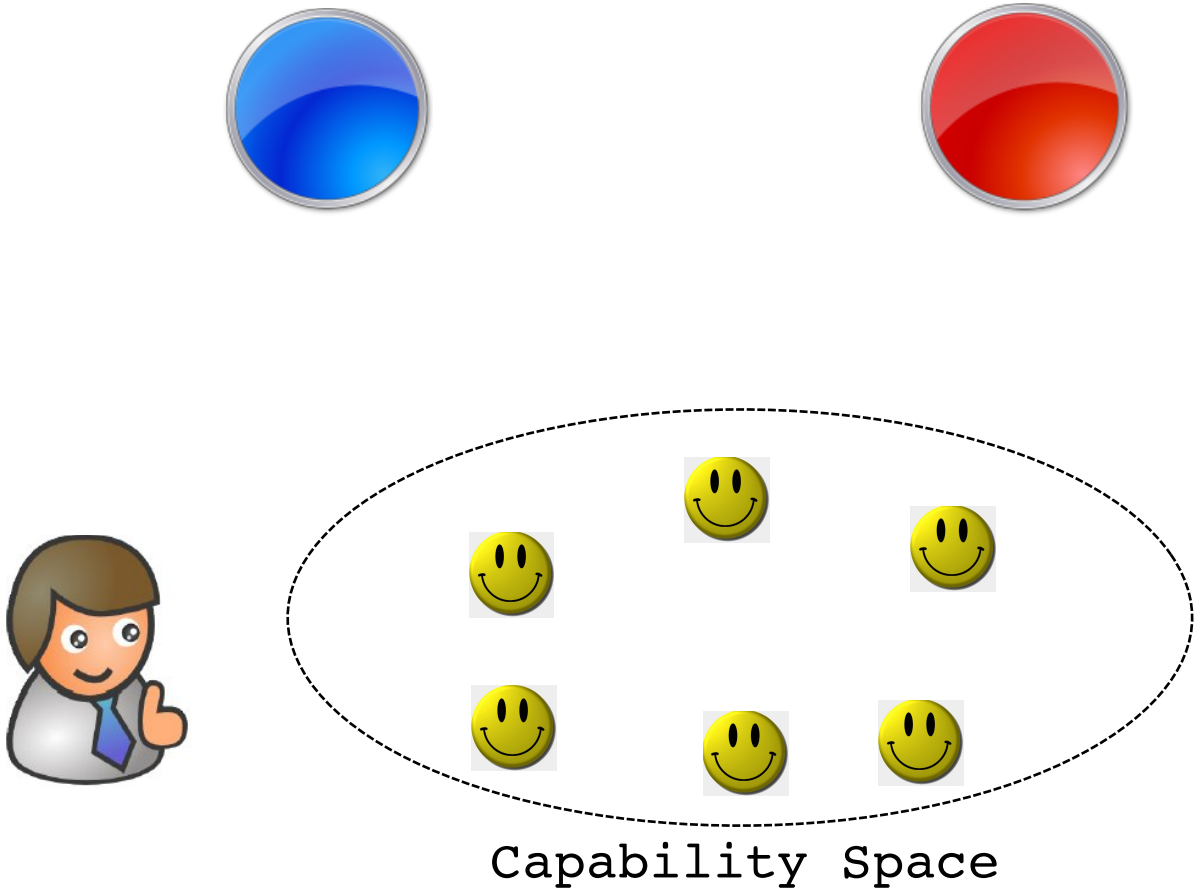
To introduce and illustrate the maximal-in-range technique, we will show a truthful $\sqrt{m}$ approximation mechanism for combinatorial allocation with subadditive valuations, in the value oracle model. The mechanism will run in time $\mathrm{poly}(n, m)$.

- Subadditivity: $v_i(S \bigcup T) \leq v_i(S) + v_i(T)$ for all $S, T \subseteq [m]$
- Value Oracle: v_i presented as a black-box which returns $v_i(S)$ on input S. Or, less generally, v_i is represented in some language such that $v_i(S)$ can be computed in $\mathrm{poly}(m)$ time.

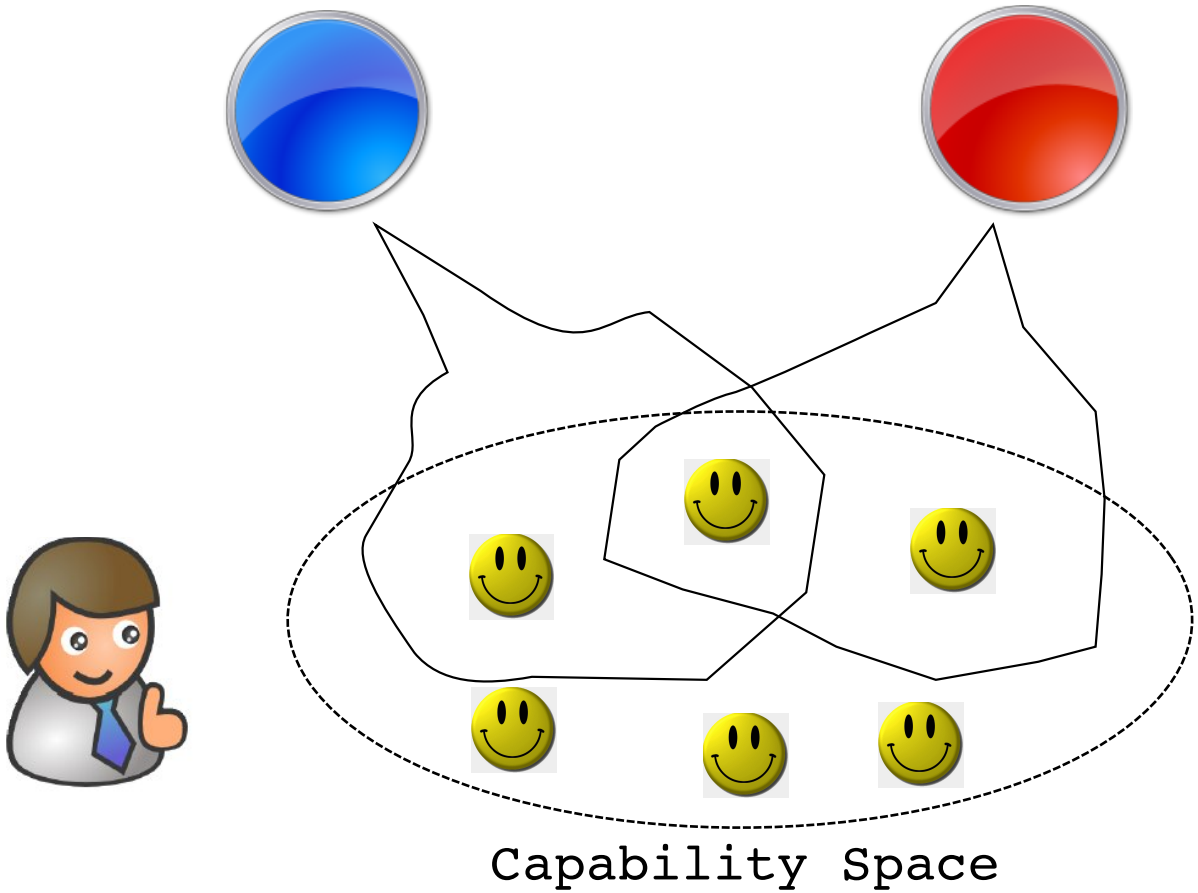
For concreteness, we fix a class of valuations that is subadditive, admits a succint representation, and for which value oracles are implementable efficiently.

Coverage Valuations

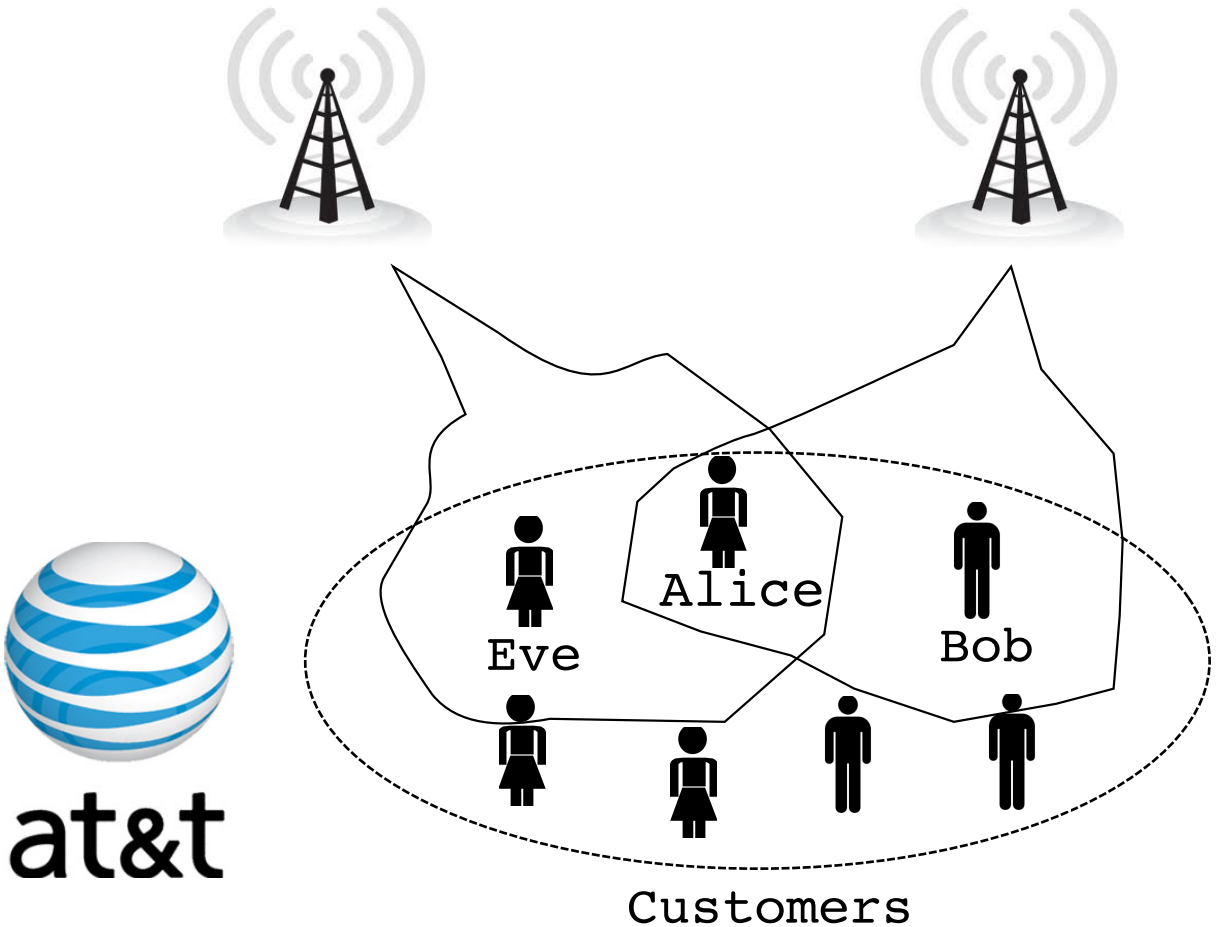
Coverage Valuations



Coverage Valuations



Coverage Valuations



Recall: The VCG Mechanism

Vickrey Clarke Groves (VCG) Mechanism for CA

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Choose allocation $(S_1, S_2, \ldots, S_n)$ maximizing $\sum_i v_i(S_i)$.
3. Charge each player i his externality

Recall: The VCG Mechanism

Vickrey Clarke Groves (VCG) Mechanism for CA

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Choose allocation $(S_1, S_2, \ldots, S_n)$ maximizing $\sum_i v_i(S_i)$.
3. Charge each player i his externality

The allocation rule can not be implemented $\mathrm{poly}(n, m)$ time, since finding a welfare maximizing allocation is NP-hard.

- Reduction from MAX-3-COLORING [Khot et al '08]

Recall: The VCG Mechanism

Vickrey Clarke Groves (VCG) Mechanism for CA

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Choose allocation $(S_1, S_2, \ldots, S_n)$ maximizing $\sum_i v_i(S_i)$.
3. Charge each player i his externality

The allocation rule can not be implemented $\mathrm{poly}(n, m)$ time, since finding a welfare maximizing allocation is NP-hard.

- Reduction from MAX-3-COLORING [Khot et al '08]

Luckily, other allocation rules can be "plugged in" to VCG while preserving truthfulness.

Maximal in Range Allocation Rules

Maximal-in-Range

AI allocation rule $f : \mathcal{V}_1 \times \ldots \times \mathcal{V}_n \to \Omega$ is maximal in range if there exists a set $\mathcal{R} \subseteq \Omega$, known as the range of f, such that

$$f(v_1, \ldots, v_n) \in \operatorname*{argmax}_{\omega \in \mathcal{R}} \sum_i v_i(\omega)$$

Maximal in Range Allocation Rules

Maximal-in-Range

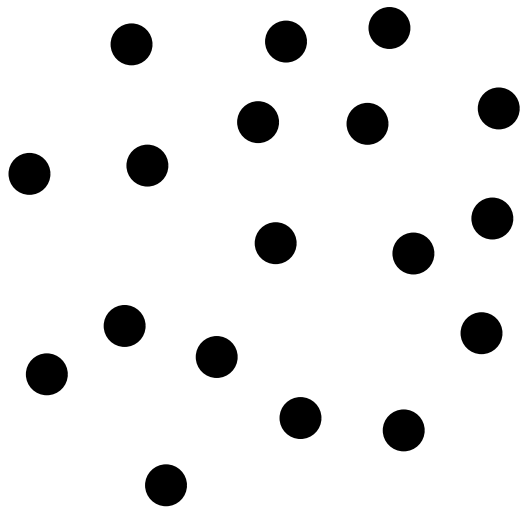
AI allocation rule $f : \mathcal{V}_1 \times \ldots \times \mathcal{V}_n \to \Omega$ is maximal in range if there exists a set $\mathcal{R} \subseteq \Omega$, known as the range of f, such that

$$f(v_1, \ldots, v_n) \in \operatorname*{argmax}_{\omega \in \mathcal{R}} \sum_i v_i(\omega)$$

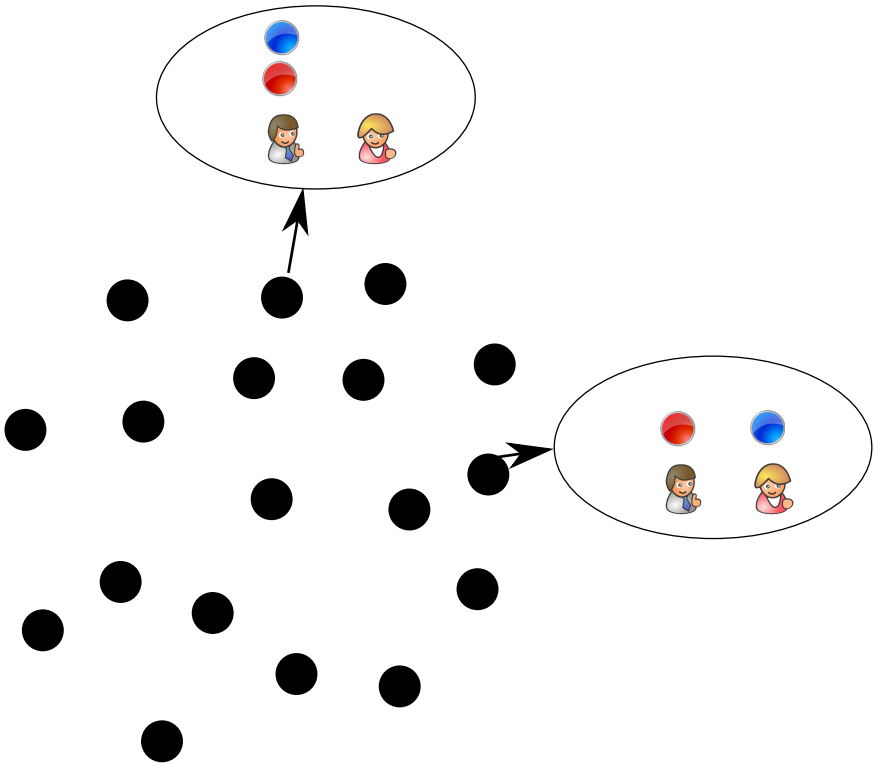
Motivation

Such an allocation rule maximizes welfare over some set of allocations $\mathcal{R}$, so remains compatible with the VCG mechanism. However, welfare maximization over $\mathcal{R}$ may be possible in polynomial time if $\mathcal{R}$ chosen properly.

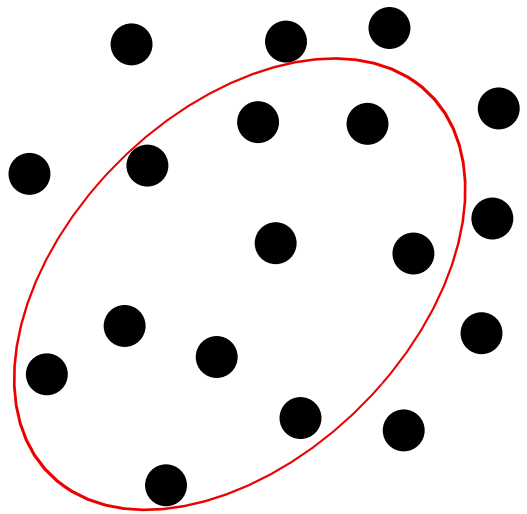
Maximal in Range Allocation Rules



Maximal in Range Allocation Rules



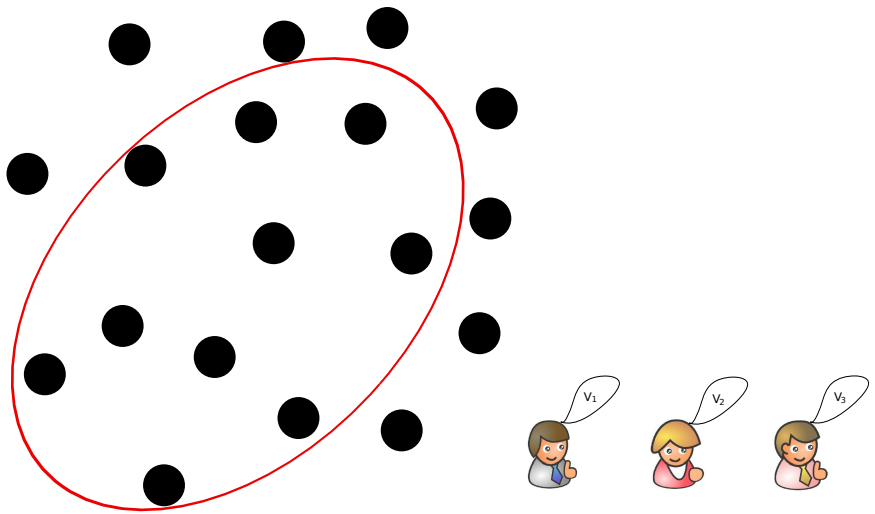
Maximal in Range Allocation Rules



Maximal in Range

1. Fix subset $\mathcal{R}$ of allocations up-front, called the range.
   - Independent of player valuations

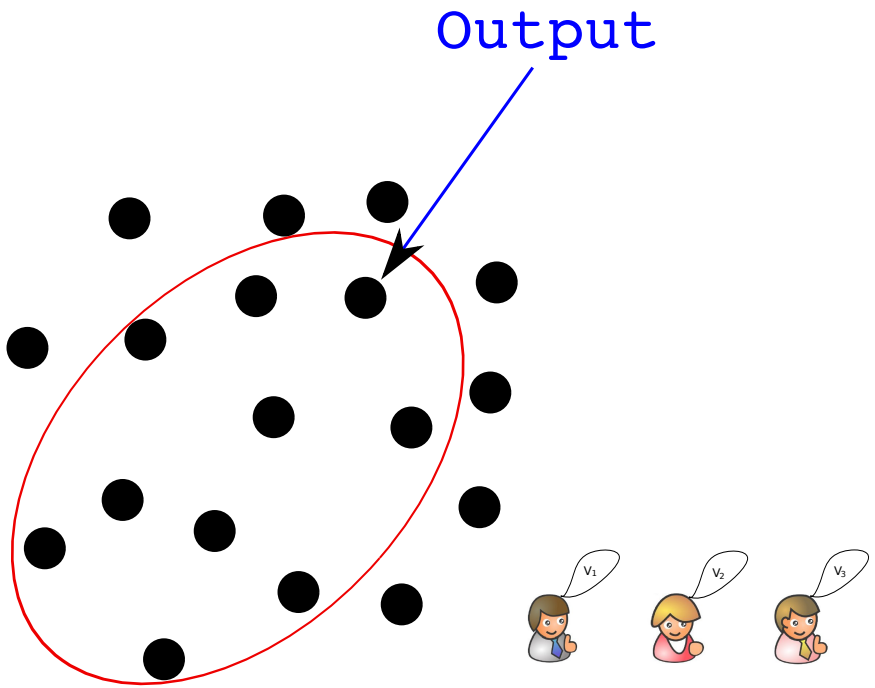
Maximal in Range Allocation Rules



Maximal in Range

1. Fix subset $\mathcal{R}$ of allocations up-front, called the range.
   - Independent of player valuations
2. Read player valuations.

Maximal in Range Allocation Rules



Maximal in Range

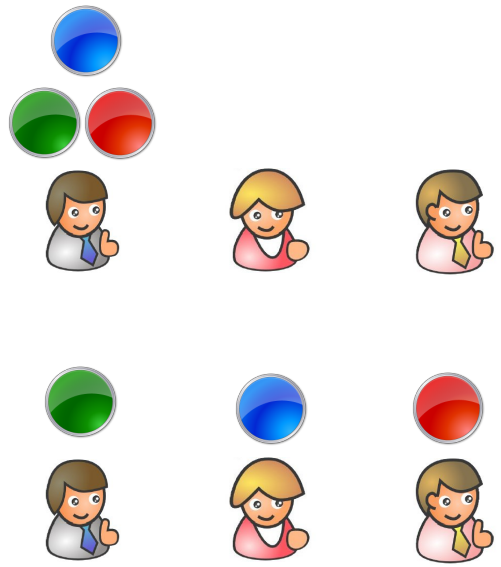
1. Fix subset $\mathcal{R}$ of allocations up-front, called the range.
   - Independent of player valuations
2. Read player valuations.
3. Output the allocation in $\mathcal{R}$ maximizing social welfare.

All-or-One Allocation Rule for CA

Consider the maximal in range allocation rule with the following range [Dobzinski et al ’05].

Range

Allocations that either allocate all items to a single player, or each player at most one item.



Maximal in Range Mechanisms

Maximal-in-Range

AI mechanism (f, p) is maximal in range if f is maximal in range for some range $\mathcal{R}$, and

$$p_i(v) = h_i(v_{-i}) - \sum_{j \neq i} v_j(f(v)).$$

Letting $h_i(v_{-i}) = \max_{\omega \in \mathcal{R}} \sum_{j \neq i} v_j(\omega)$ be the Clarke pivot relative to $\mathcal{R}$ gives the same properties as the Clarke pivot in the VCG mechanism.

Maximal in Range Mechanisms

Maximal in Range Mechanism for CA

For a fixed range $\mathcal{R} \subseteq \Omega$, chosen independently of v_i's

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Choose allocation $(S_1, S_2, \ldots, S_n) \in \mathcal{R}$ maximizing $\sum_i v_i(S_i)$.
3. Charge each player i his externality relative to $\mathcal{R}$
   - $p_i(v) = \max_{(T_1,\ldots,T_n)\in\mathcal{R}} \sum_{j\neq i} v_j(\omega) - \sum_{j\neq i} v_j(S_j)$

Maximal in Range Mechanisms

Maximal in Range Mechanism for CA

For a fixed range $\mathcal{R} \subseteq \Omega$, chosen independently of v_i's

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Choose allocation $(S_1, S_2, \ldots, S_n) \in \mathcal{R}$ maximizing $\sum_i v_i(S_i)$.
3. Charge each player i his externality relative to $\mathcal{R}$
   - $p_i(v) = \max_{(T_1,\ldots,T_n)\in\mathcal{R}} \sum_{j\neq i} v_j(\omega) - \sum_{j\neq i} v_j(S_j)$

Fact

Every maximal in range algorithm is truthful.

proof

Simply VCG applied to a "smaller" mechanism design problem, namely that where the set of allocations is $\mathcal{R}$ rather than Ω.

Maximal in Range Mechanisms

Maximal in Range Mechanism for CA

For a fixed range $\mathcal{R} \subseteq \Omega$, chosen independently of v_i's

1. Solicit report $v_i \in \mathcal{V}_i$ from each player i
2. Choose allocation $(S_1, S_2, \ldots, S_n) \in \mathcal{R}$ maximizing $\sum_i v_i(S_i)$.
3. Charge each player i his externality relative to $\mathcal{R}$
   - $p_i(v) = \max_{(T_1,\ldots,T_n)\in\mathcal{R}} \sum_{j\neq i} v_j(\omega) - \sum_{j\neq i} v_j(S_j)$

Upshot

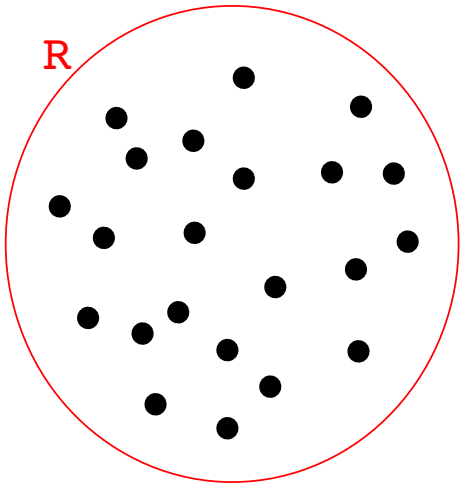
We have reduced design of a truthful polynomial-time mechanism to designing an polynomial-time allocation rule (i.e. approximation algorithm) that is maximal-in-range.

Designing MIR Algorithms

- A good MIR allocation rule achieves a good “trade-off” between approximation ratio, and runtime

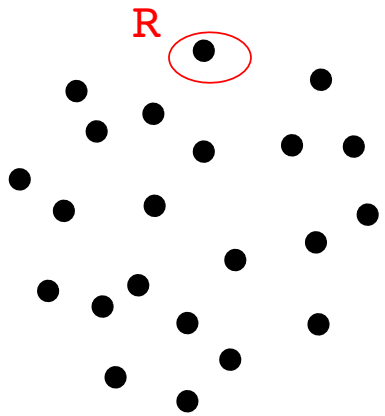
Designing MIR Algorithms

- A good MIR allocation rule achieves a good "trade-off" between approximation ratio, and runtime
- At one extreme: $\mathcal{R} =$ all allocations.
  - Approximation ratio = 1
  - NP-hard if the problem is NP-hard



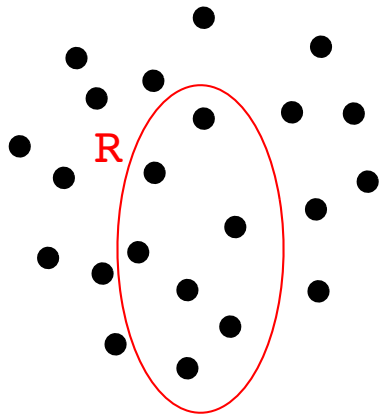
Designing MIR Algorithms

- A good MIR allocation rule achieves a good "trade-off" between approximation ratio, and runtime
- At one extreme: $\mathcal{R} =$ all allocations.
  - Approximation ratio = 1
  - NP-hard if the problem is NP-hard
- At another extreme: $\mathcal{R} = \{x\}$ a singleton
  - Definitely polytime
  - Approximation ratio is terrible



Designing MIR Algorithms

- A good MIR allocation rule achieves a good "trade-off" between approximation ratio, and runtime
- At one extreme: $\mathcal{R} =$ all allocations.
  - Approximation ratio = 1
  - NP-hard if the problem is NP-hard
- At another extreme: $\mathcal{R} = \{x\}$ a singleton
  - Definitely polytime
  - Approximation ratio is terrible

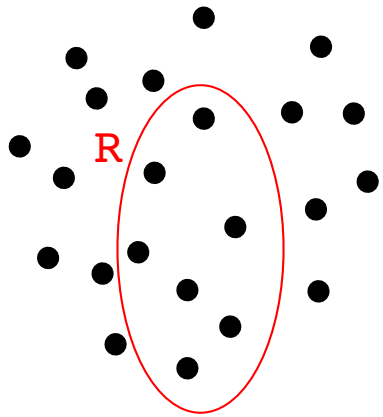


Is there a "sweet spot"?

- Large enough for good approximation
- Small/well-structured enough for polytime optimization

Designing MIR Algorithms

- A good MIR allocation rule achieves a good "trade-off" between approximation ratio, and runtime
- At one extreme: $\mathcal{R} =$ all allocations.
  - Approximation ratio = 1
  - NP-hard if the problem is NP-hard
- At another extreme: $\mathcal{R} = \{x\}$ a singleton
  - Definitely polytime
  - Approximation ratio is terrible



Is there a "sweet spot"?

- Large enough for good approximation
- Small/well-structured enough for polytime optimization

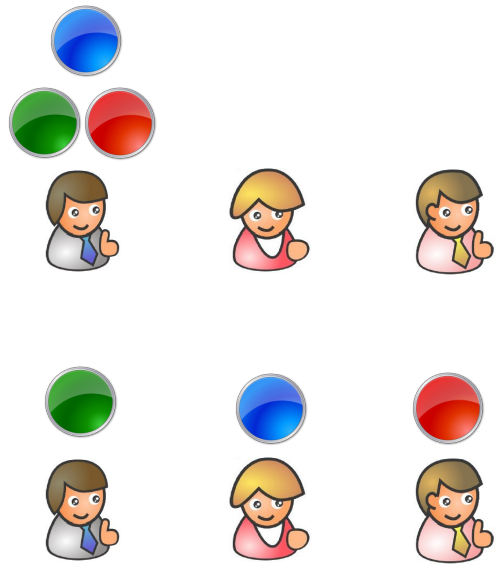
The design of a maximal in range algorithm is akin to algorithm design in a restricted computational model.

Recall: All-or-One Allocation Rule for CA

Consider the maximal in range allocation rule with the following range [Dobzinski et al '05].

Range

Allocations that either allocate all items to a single player, or each player at most one item.



Proof of Polynomial-time Implementability

Lemma

The all-or-one allocation rule can be implemented in $\mathrm{poly}(n, m)$ *time.*

Proof

- Find the best allocation of all items to one player by evaluating the welfare of n allocations.
- Find the best allocation of at most one item per player by solving a bipartite maximum matching problem with the n players on one side, and the m items on the other.
- Output the better of the two.

Proof of Approximation

Lemma

The all-or-one allocation rule is a $O(\sqrt{m})$ approximation when players have coverage valuations.

Proof of Approximation

Lemma

The all-or-one allocation rule is a $O(\sqrt{m})$ approximation when players have coverage valuations.

Proof

- Fix coverage valuations $v_1, \ldots, v_n$.
- Let $(S_1^*, \ldots, S_n^*)$ be welfare-maximizing allocation, with welfare $OPT = \sum_i v_i(S_i^*)$
- Suffices to show that there is an all-or-one allocation with welfare at least $\frac{1}{O(\sqrt{m})} OPT$.

Proof of Approximation

Lemma

The all-or-one allocation rule is a $O(\sqrt{m})$ approximation when players have coverage valuations.

Proof

- Fix coverage valuations $v_1, \ldots, v_n$.
- Let $(S_1^*, \ldots, S_n^*)$ be welfare-maximizing allocation, with welfare $OPT = \sum_i v_i(S_i^*)$
- Suffices to show that there is an all-or-one allocation with welfare at least $\frac{1}{O(\sqrt{m})} OPT$.
- Two cases:
  1. Players i with $|S_i^*| \geq \sqrt{m}$ account for at least half the welfare of S^*: Since there are at most $\sqrt{m}$ such players, at least one player accounts for $\frac{1}{2\sqrt{m}} OPT$. The allocation awarding all items to this player has welfa at least that much.

Proof of Approximation

Lemma

The all-or-one allocation rule is a $O(\sqrt{m})$ approximation when players have coverage valuations.

Proof

- Fix coverage valuations $v_1, \ldots, v_n$.
- Let $(S_1^*, \ldots, S_n^*)$ be welfare-maximizing allocation, with welfare $OPT = \sum_i v_i(S_i^*)$
- Suffices to show that there is an all-or-one allocation with welfare at least $\frac{1}{O(\sqrt{m})} OPT$.
- Two cases:
  2. Players i with $|S_i^*| \leq \sqrt{m}$ account for at least half the welfare of S^*: Fo each such player i, there is a single item $j_i \in S_i^*$ with $v_i(\{j_i\}) \geq v_i(S_i^*)/\sqrt{m}$. Namely, let j_i be the item in S_i^* covering the mo capabilities. The allocation awarding only j_i to each such player i has value at least $\frac{OPT}{2\sqrt{m}}$.

The maximal-in-range mechanism with the all-or-one allocation rule is a $O(\sqrt{m})$ approximation, and runs in time $\mathrm{poly}(n, m)$.

Theorem

There is a truthful, $O(\sqrt{m})$-approximate mechanism for combinatorial allocation with coverage valuatoins, which runs in $\mathrm{poly}(n, m)$ time.

The maximal-in-range mechanism with the all-or-one allocation rule is a $O(\sqrt{m})$ approximation, and runs in time $\mathrm{poly}(n, m)$.

Theorem

There is a truthful, $O(\sqrt{m})$-approximate mechanism for combinatorial allocation with coverage valuatoins, which runs in $\mathrm{poly}(n, m)$ time.

Note: Applies more generally to subadditive valuations that admit a value oracle.

Next Time

- Characterizations of Dominant-Strategy Incentive Compatibility
- Maximal in Distributional Range Algorithms
- Tha Lavi-Swamy Linear-programming technique