CS599: Algorithm Design in Strategic Settings
Fall 2012
Lecture 8: Prior-Free Multi-Parameter Mechanism Design (Continued)

Instructor: Shaddin Dughmi

Outline

1. Review

2. Maximal in Distributional Range Algorithms

3. The Lavi Swamy Linear Programming Approach

Outline

1. Review
2. Maximal in Distributional Range Algorithms
3. The Lavi Swamy Linear Programming Approach

Recall: Mechanism Design Problem in Quasi-linear Settings

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 $v_i : T_i \times \Omega \to \mathbb{R}$

Recall: Mechanism Design Problem in Quasi-linear Settings

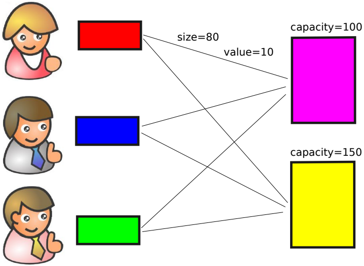
Public (common knowledge) inputs describes

- 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.

Example: Generalized Assignment

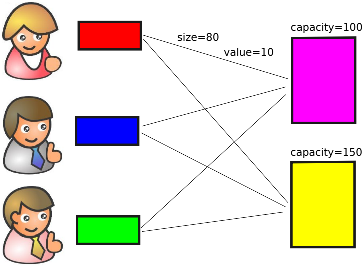


- n self-interested agents (the players), m machines.
- s_{ij} is the size of player i's task on machine j. (public)
- C_j is machine j's capacity. (public)
- $v_i(j)$ is player i's value for his task going on machine j. (private)

Goal

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

Example: Generalized Assignment



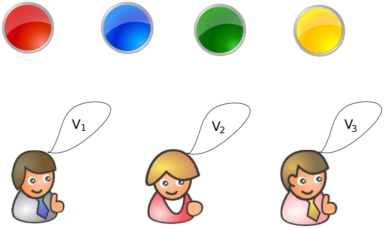
- n self-interested agents (the players), m machines.
- s_{ij} is the size of player i's task on machine j. (public)
- C_j is machine j's capacity. (public)
- $v_i(j)$ is player i's value for his task going on machine j. (private)

Goal

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

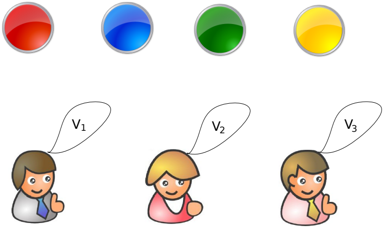
Note: When single machine, this is knapsack allocation.

Example: 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

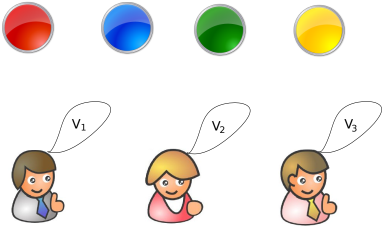


- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

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 $\to \mathbb{R}$.
  - $v_i(S)$ 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 consider families of restricted valuations with a succinct representation.

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 $f(v_1, \dots, v_n)$
3. Charge each player i payment $p_i(v_1, \dots, v_n)$

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.

Recall: 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.

Recall: Vickrey Clarke Groves (VCG) Mechanism with Clarke Pivot

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

Recall: Vickrey Clarke Groves (VCG) Mechanism with Clarke Pivot

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

Theorem

VCG is dominant-strategy truthful. Moreover, when using the Clarke pivot, it is individually rational for problems with nonnegative valuations and payments are nonnegative.

Applications: matching, sponsored search, routing, and many more.

Recall: Vickrey Clarke Groves (VCG) Mechanism with Clarke Pivot

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

Theorem

VCG is dominant-strategy truthful. Moreover, when using the Clarke pivot, it is individually rational for problems with nonnegative valuations and payments are nonnegative.

Applications: matching, sponsored search, routing, and many more.

Bad News

Requires exact solution of welfare maximization problem, which is infeasible in many settings.

E.g. Combinatorial allocation, Generalized assignment, ...

Recall: Maximal in Range Allocation Rules

Maximal-in-Range

Al 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

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

Recall: Maximal in Range Allocation Rules

Maximal-in-Range

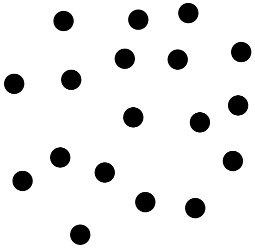
Al allocation rule $f : \mathcal{V}_1 \times \ldots \times \mathcal{V}_n \to \Omega$ is maximal in range if there exists a set $\mathcal{R} \subseteq \Omega$, known as the range of f, such that

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

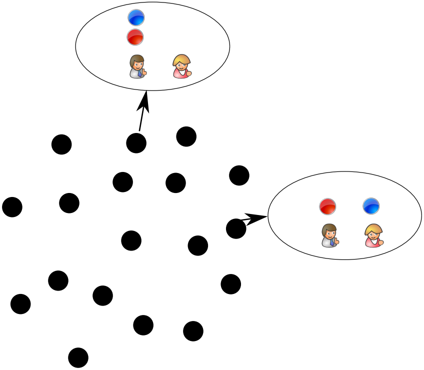
Motivation

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.

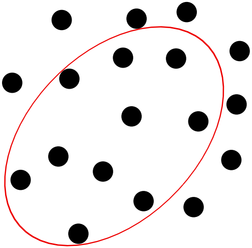
Recall: Maximal in Range Allocation Rules



Recall: Maximal in Range Allocation Rules



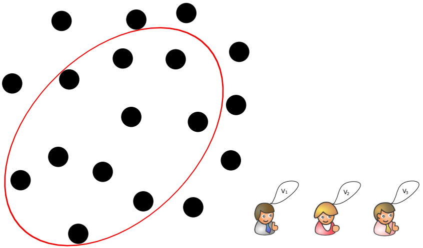
Recall: Maximal in Range Allocation Rules



Maximal in Range

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

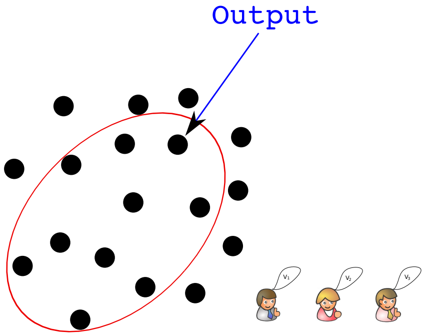
Recall: Maximal in Range Allocation Rules



Maximal in Range

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

Recall: 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.

Recall: Maximal in Range Allocation Rules

Fact

For any mechanism design problem, every maximal in range allocation rule is implementable in dominant-strategies by plugging into VCG. Moreover, if the maximal in range algorithm runs in polynomial time, then so does the resulting dominant-strategy truthful mechanism.

Upshot

For NP-hard welfare maximization mechanism design problems (such as GAP, CA, and others), this reduces the design of dominant-strategy truthful, polynomial-time mechanisms to the design of a polynomial-time maximal-in-range allocation algorithms with the desired approximation ratio.

Last Time: Maximal-in-Range Mechanism for Combinatorial Allocation

- We considered combinatorial allocation with coverage valuations.
- We saw the all-or-one allocation rule for this problem
  - Polynomial time
  - $O(\sqrt{m})$ approximation algorithm for maximizing welfare
- Concluded: There is a $O(\sqrt{m})$-approximate, polynomial-time, dominant-strategy truthful mechanism for welfare maximization in this problem.

Last Time: Maximal-in-Range Mechanism for Combinatorial Allocation

- We considered combinatorial allocation with coverage valuations.
- We saw the all-or-one allocation rule for this problem
  - Polynomial time
  - $O(\sqrt{m})$ approximation algorithm for maximizing welfare
- Concluded: There is a $O(\sqrt{m})$-approximate, polynomial-time, dominant-strategy truthful mechanism for welfare maximization in this problem.

The maximal-in-range approach has only taken researchers so far. More general ideas were necessary to obtain improved results for more multi-parameter problems.

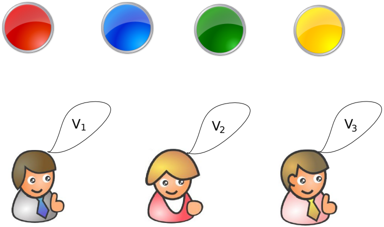
Coming Up Today

- Maximal in Distributional Range Algorithms
- Tha Lavi-Swamy Linear-programming technique

Outline

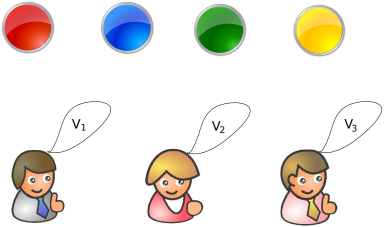
1. Review
2. Maximal in Distributional Range Algorithms
3. The Lavi Swamy Linear Programming Approach

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

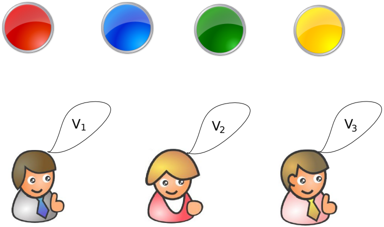


- n players, m items.
- Private valuation v_i : set of items $\rightarrow \mathbb{R}$.
  - $v_i(S)$ is player i's value for bundle S.

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.

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

Recall

Using a maximal in range allocation algorithm in lieu of an optimal allocation algorithm preserves truthfulness, and can in some cases recover polynomial time.

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

Using a maximal in range allocation algorithm in lieu of an optimal allocation algorithm preserves truthfulness, and can in some cases recover polynomial time.

The same is true for a randomized generalization of MIR, which appears more powerful.

Maximal in Distributional Range Allocation Rules

Maximal-in-Distributional-Range (MIDR)

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

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

Maximal in Distributional Range Allocation Rules

Maximal-in-Distributional-Range (MIDR)

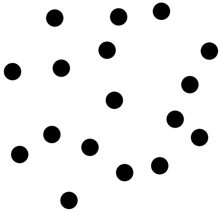
Al allocation rule $f : \mathcal{V}_1 \times \ldots \times \mathcal{V}_n \to \Omega$ is maximal in distributional range if there exists a set $\mathcal{R} \subseteq \Delta(\Omega)$, known as the distributional range of f, such that

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

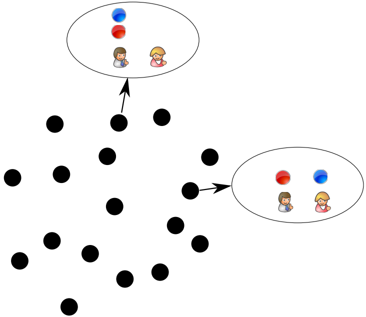
In Other Words

Such an allocation rule samples a distribution in $\mathcal{R}$ maximizing expected social welfare. Maximal in range allocation rules are the special case of MIDR when $\mathcal{R}$ is a family of point distributions.

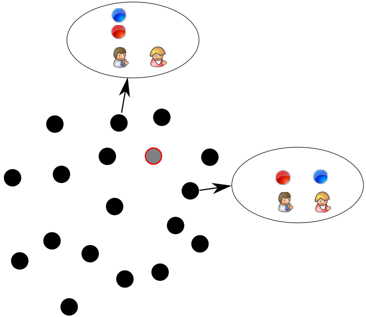
Maximal in Distributional Range Allocation Rules



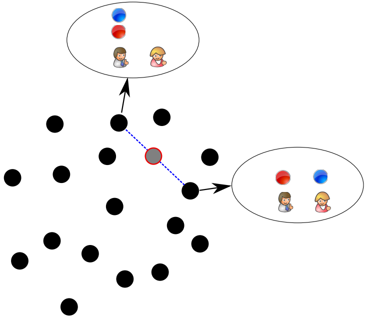
Maximal in Distributional Range Allocation Rules



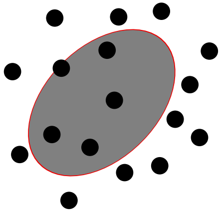
Maximal in Distributional Range Allocation Rules



Maximal in Distributional Range Allocation Rules



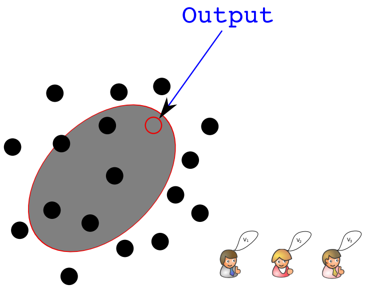
Maximal in Distributional Range Allocation Rules



Maximal in Distributional Range

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

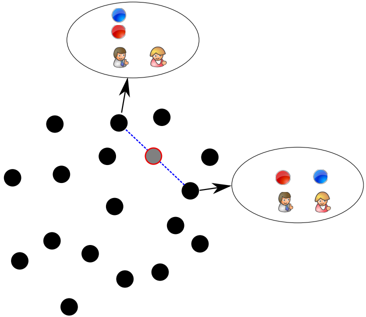
Maximal in Distributional Range Allocation Rules



Maximal in Distributional Range

1. Fix subset $\mathcal{R}$ of distributions over allocations up-front, called the distributional range.
   - Independent of player valuations
2. Given player values, find the distribution in $\mathcal{R}$ maximizing expected social welfare.

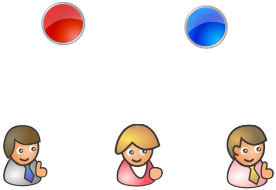
Maximal in Distributional Range Allocation Rules



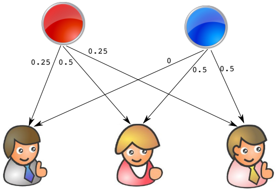
Maximal in Distributional Range

1. Fix subset $\mathcal{R}$ of distributions over allocations up-front, called the distributional range.
   - Independent of player valuations
2. Given player values, find the distribution in $\mathcal{R}$ maximizing expected social welfare.
3. Sample this distribution

Example of MIDR

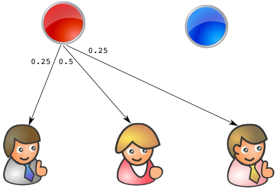


Example of MIDR



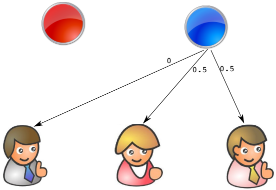
- Independent lottery:
  - Associates with each player i and item j probability x_{ij} that i gets j
  - Each item j assigned independently with those probabilities.

Example of MIDR



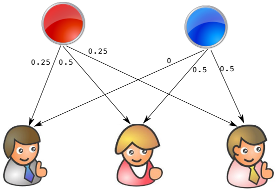
- Independent lottery:
  - Associates with each player i and item j probability x_{ij} that i gets j
  - Each item j assigned independently with those probabilities.

Example of MIDR



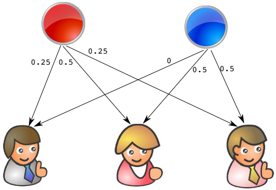
- Independent lottery:
  - Associates with each player i and item j probability x_{ij} that i gets j
  - Each item j assigned independently with those probabilities.

Example of MIDR



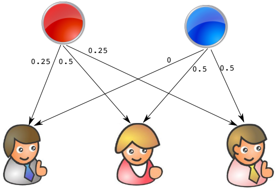
- Independent lottery:
  - Associates with each player i and item j probability x_{ij} that i gets j
  - Each item j assigned independently with those probabilities.
- Each set of fractions x_{ij} defines a different independent lottery
- The set of independent lotteries is a distributional range.

Example of MIDR



- Independent lottery:
  - Associates with each player i and item j probability x_{ij} that i gets j
  - Each item j assigned independently with those probabilities.
- Each set of fractions x_{ij} defines a different independent lottery
- The set of independent lotteries is a distributional range.
- Easy Fact: MIDR over all independent lotteries is as computationally hard as exact optimization over all allocations.

Example of MIDR



- Independent lottery:
  - Associates with each player i and item j probability x_{ij} that i gets j
  - Each item j assigned independently with those probabilities.
- Each set of fractions x_{ij} defines a different independent lottery
- The set of independent lotteries is a distributional range.
- Easy Fact: MIDR over all independent lotteries is as computationally hard as exact optimization over all allocations.

Next lecture, we use range of independent lotteries where each $x_{ij} \leq 0.63$ to improve last lecture's result of $\sqrt{m}$ approximation to a 0.63 approximation.

Maximal-in-Distributional-Range Mechanism

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

$$\mathbf{E}[p_i(v)] = h_i(v_{-i}) - \mathbf{E}\left[\sum_{j \neq i} v_j(f(v))\right].$$

Fact

- Every maximal in distributional range mechanism is truthful.
- When $h_i(v_{-i}) = \max_{D \in \mathcal{R}} \mathbf{E}_{\omega \sim D}[\sum_{j \neq i} v_j(\omega)]$ is the Clarke pivot relative to $\mathcal{R}$, the mechanism is individually rational in expectation (when valuations are nonnegative), and expected payments are nonnegative.

Maximal-in-Distributional-Range Mechanism

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

$$\mathbf{E}[p_i(v)] = h_i(v_{-i}) - \mathbf{E}\left[\sum_{j \neq i} v_j(f(v))\right].$$

Fact

- Every maximal in distributional range mechanism is truthful.
- When $h_i(v_{-i}) = \max_{D \in \mathcal{R}} \mathbf{E}_{\omega \sim D}[\sum_{j \neq i} v_j(\omega)]$ is the Clarke pivot relative to $\mathcal{R}$, the mechanism is individually rational in expectation (when valuations are nonnegative), and expected payments are nonnegative.

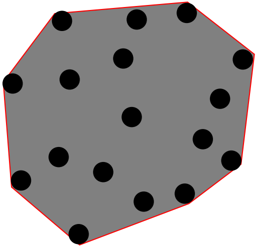
Easy exercise: Given black-box access to f, can sample payments satisfying both desiderata using $n + 1$ calls to f.

Designing MIDR Algorithms

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

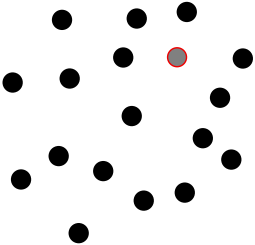
Designing MIDR Algorithms

- A good MIDR allocation algorithm achieves a good “trade-off” between approximation ratio, and runtime
- At one extreme: $\mathcal{R} =$ all distributions
  - Approximation ratio = 1
  - NP-hard if the problem is NP-hard



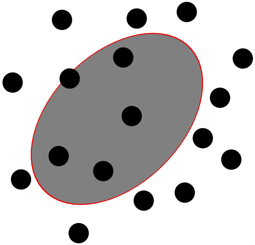
Designing MIDR Algorithms

- A good MIDR allocation algorithm achieves a good “trade-off” between approximation ratio, and runtime
- At one extreme: $\mathcal{R} =$ all distributions
  - 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 MIDR Algorithms

- A good MIDR allocation algorithm achieves a good “trade-off” between approximation ratio, and runtime
- At one extreme: $\mathcal{R}$ = all distributions
  - 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



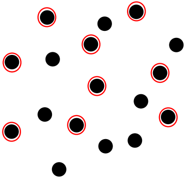
Is there a “sweet spot”?

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

Intuition: Why Randomness Helps

- Limited successes using Maximal in Range
- For some problems, researchers showed that any set of discrete allocations large enough for a good approximation is “complex” enough to be NP-hard.

Discrete problems tend to be computationally hard!

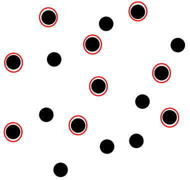


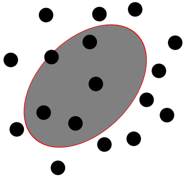
Intuition: Why Randomness Helps

- Limited successes using Maximal in Range
- For some problems, researchers showed that any set of discrete allocations large enough for a good approximation is "complex" enough to be NP-hard.

Discrete problems tend to be computationally hard!

- Intuition from linear programming and convex optimization suggests that optimization over convex/smooth sets is easier...
- Distributional ranges can be both "large" and "nice" (smooth/convex).





Outline

1. Review
2. Maximal in Distributional Range Algorithms
3. The Lavi Swamy Linear Programming Approach

Overview

- Considers welfare maximization mechanism design problems.
- Reduces the design of polynomial-time MIDR mechanisms to the design of linear programming relaxations satisfying certain conditions.

Overview

- Considers welfare maximization mechanism design problems.
- Reduces the design of polynomial-time MIDR mechanisms to the design of linear programming relaxations satisfying certain conditions.

Theorem (Lavi and Swamy)

Consider a welfare-maximization mechanism design problem. If

- *the problem can be written as a packing integer linear program with integrality gap at most α,*

Overview

- Considers welfare maximization mechanism design problems.
- Reduces the design of polynomial-time MIDR mechanisms to the design of linear programming relaxations satisfying certain conditions.

Theorem (Lavi and Swamy)

Consider a welfare-maximization mechanism design problem. If

- *the problem can be written as a packing integer linear program with integrality gap at most α,*
- *the relaxation of the PILP can be solved in polynomial time,*

Overview

- Considers welfare maximization mechanism design problems.
- Reduces the design of polynomial-time MIDR mechanisms to the design of linear programming relaxations satisfying certain conditions.

Theorem (Lavi and Swamy)

Consider a welfare-maximization mechanism design problem. If

- *the problem can be written as a packing integer linear program with integrality gap at most α,*
- *the relaxation of the PILP can be solved in polynomial time,*
- *and there is an algorithm that shows integrality gap α,*

Overview

- Considers welfare maximization mechanism design problems.
- Reduces the design of polynomial-time MIDR mechanisms to the design of linear programming relaxations satisfying certain conditions.

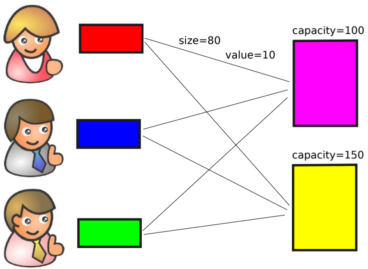
Theorem (Lavi and Swamy)

Consider a welfare-maximization mechanism design problem. If

- *the problem can be written as a packing integer linear program with integrality gap at most α,*
- *the relaxation of the PILP can be solved in polynomial time,*
- *and there is an algorithm that shows integrality gap α,*

then an α-approximate MIDR algorithm can be generically derived in polynomial time.

Recall: Generalized Assignment

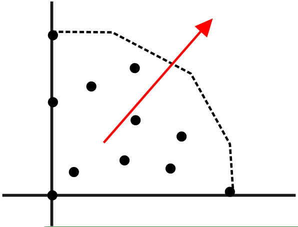


- n self-interested agents (the players), m machines.
- s_{ij} is the size of player i's task on machine j. (public)
- C_j is machine j's capacity. (public)
- $v_i(j)$ is player i's value for his task going on machine j. (private)

Goal

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

Packing Linear Integer Programs



Generic PILP
$(A, b, v \geq 0)$

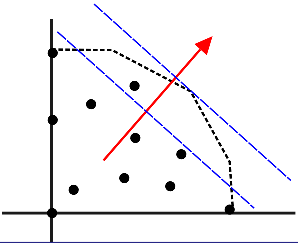
$$\begin{array}{ll} \text{max} & \sum_i v_i^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^m \end{array}$$

Example: GAP PILP

$$\begin{array}{lll} \text{max} & \sum_{ij} v_i(j) x_{ij} & \\ \text{s.t.} & \sum_i s_{ij} x_{ij} \leq C_j, & \text{for } j \in [m]. \\ & x_{ij} \leq 1, & \text{for } i \in [n], j \in [m]. \\ & x_{ij} \geq 0, & \text{for } i \in [n], j \in [m]. \\ & x_{ij} \in \{0, 1\}, & \text{for } i \in [n], j \in [m]. \end{array}$$

Removing the integrality constraint gives a linear programming relaxation of the problem.

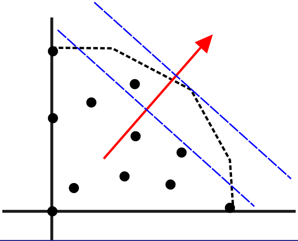
Packing Linear Integer Programs



Definition (Integrality Gap)

A PILP has integrality gap at most α if, for every objective $v \in \mathbb{R}^m_+$, the ratio of the welfare of the best fractional solution and the best integer solution is at most α.

Packing Linear Integer Programs



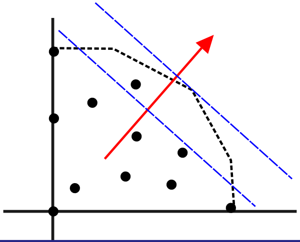
Definition (Integrality Gap)

A PILP has integrality gap at most α if, for every objective $v \in \mathbb{R}_+^m$, the ratio of the welfare of the best fractional solution and the best integer solution is at most α.

Theorem [Shmoys/Tardos '93, Chekuri/Khanna '05]

GAP PILP has integrality gap at most 2.

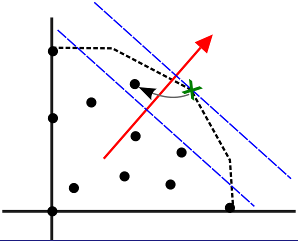
Packing Linear Integer Programs



Definition

An algorithm for a PILP shows an integrality gap of α if, for every objective $v \in \mathbb{R}_+^m$, it always outputs an integer solution with objective value at least $1/\alpha$ of that of the best fractional solution, in expectation.

Packing Linear Integer Programs

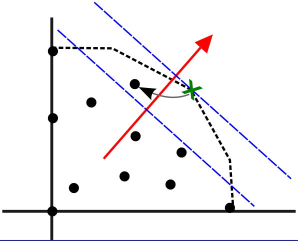


Definition

An algorithm for a PILP shows an integrality gap of α if, for every objective $v \in \mathbb{R}_+^m$, it always outputs an integer solution with objective value at least $1/\alpha$ of that of the best fractional solution, in expectation.

Commonly, such an algorithm “rounds” the optimal fractional solution of the LP, but this is not necessary.

Packing Linear Integer Programs



Definition

An algorithm for a PILP shows an integrality gap of α if, for every objective $v \in \mathbb{R}_+^m$, it always outputs an integer solution with objective value at least $1/\alpha$ of that of the best fractional solution, in expectation.

Theorem [Shmoys/Tardos '93, Chekuri/Khanna '05]

There is an algorithm for GAP that shows an integrality gap of 2 with respect to GAP PILP.

The algorithm rounds a fractional solution to the LP relaxation.

Theorem [Shmoys/Tardos '93, Chekuri/Khanna '05]

There is an algorithm for GAP that shows an integrality gap of 2 with respect to GAP PILP.

Theorem [Shmoys/Tardos '93, Chekuri/Khanna '05]

There is an algorithm for GAP that shows an integrality gap of 2 with respect to GAP PILP.

Observe

The relaxed GAP PILP is simply a linear program with a polynomial number of constraints, and therefore can be solved in polynomial time by the ellipsoid method, interior point methods, etc...

Therefore, the conditions for applying the Lavi Swamy framework are satisfied, yielding a polynomial-time, 2-approximate MIDR algorithm for GAP, and therefore also a polynomial-time 2-approximate truthful mechanism.

Theorem [Shmoys/Tardos '93, Chekuri/Khanna '05]

There is an algorithm for GAP that shows an integrality gap of 2 with respect to GAP PILP.

Proof of a special case

Suffices to show how to convert, in polynomial time, a fractional assignment to an integral assignment with at least half the welfare. We will prove this in the special case where $s_{ij} = s_{ik}$ for all i, j, k. For the general case, see the papers.

- Let x be fractional assignment.
- Draw bipartite graph G connecting a task to a machine if assigned fractionally
- G is a union of maximal paths and cycles

Theorem [Shmoys/Tardos '93, Chekuri/Khanna '05]

There is an algorithm for GAP that shows an integrality gap of 2 with respect to GAP PILP.

Proof of a special case

- While G has a cycle C
  - "Shift" job mass around C in whichever direction does not decrease welfare of fractional solution, until some job j is entirely removed from some machine i
  - Remove edges that no longer correspond to fractional assignments (must include (j, i))
- While G is non-empty
  - Pick a maximal path P
  - "Shift" job mass along whichever direction of P does not decrease welfare of fractional solution, until some job j is entirely removed from some machine i
  - Remove edges that no longer correspond to fractional assignments (must include (j, i))
  - Note: Machine at end P (with one fractional job) may overflow

Theorem [Shmoys/Tardos '93, Chekuri/Khanna '05]

There is an algorithm for GAP that shows an integrality gap of 2 with respect to GAP PILP.

Proof of a special case

- At the end of this process, we have an integral assignment with welfare at least that of the fractional assignment we started with, though some machines have overflowed by at most one job.
- Restore feasibility: For each machine, either toss away the overflow job or everything else, whichever guarantees half the value.

Recall: Theorem Statement

Theorem (Lavi and Swamy)

Consider a welfare-maximization problem. If

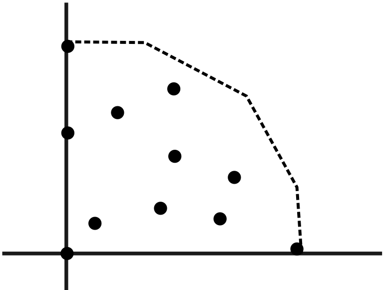
- *the problem can be written as a packing integer linear program with integrality gap at most α,*
- *the relaxation of the PILP can be solved in polynomial time,*
- *and there is an algorithm that shows integrality gap α,*

then an α-approximate MIDR algorithm can be generically derived in polynomial time.

Algorithm Outline

MIDR α-approximate Algorithm

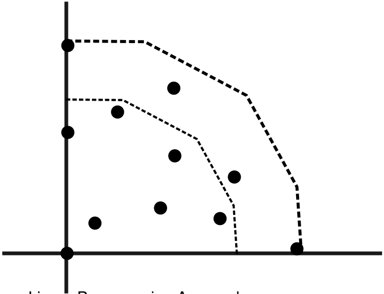
1. Scale the feasible region P of the LP relaxation down by the integrality gap α.
2. Find the optimal solution x of the scaled LP.
3. Let D_x be a distribution over integer solutions with expectation x.
4. Output a sample from D_x.



Algorithm Outline

MIDR α-approximate Algorithm

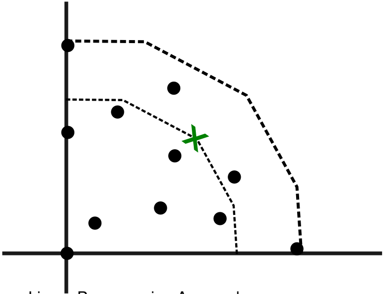
1. Scale the feasible region P of the LP relaxation down by the integrality gap α.
2. Find the optimal solution x of the scaled LP.
3. Let D_x be a distribution over integer solutions with expectation x.
4. Output a sample from D_x.



Algorithm Outline

MIDR α-approximate Algorithm

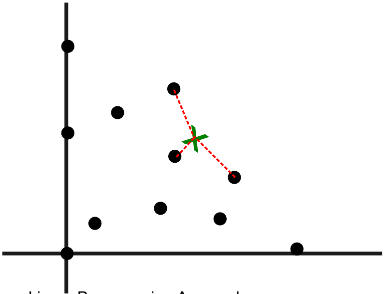
1. Scale the feasible region P of the LP relaxation down by the integrality gap α.
2. Find the optimal solution x of the scaled LP.
3. Let D_x be a distribution over integer solutions with expectation x.
4. Output a sample from D_x.



Algorithm Outline

MIDR α-approximate Algorithm

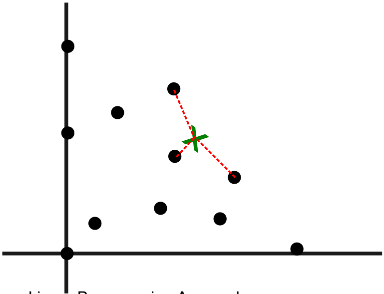
1. Scale the feasible region P of the LP relaxation down by the integrality gap α.
2. Find the optimal solution x of the scaled LP.
3. Let D_x be a distribution over integer solutions with expectation x.
4. Output a sample from D_x.



Algorithm Outline

MIDR α-approximate Algorithm

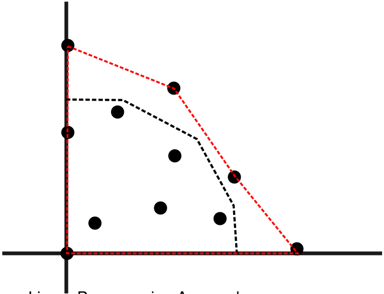
1. Scale the feasible region P of the LP relaxation down by the integrality gap α.
2. Find the optimal solution x of the scaled LP.
3. Let D_x be a distribution over integer solutions with expectation x.
4. Output a sample from D_x.



Algorithm Outline

MIDR α-approximate Algorithm

1. Scale the feasible region P of the LP relaxation down by the integrality gap α.
2. Find the optimal solution x of the scaled LP.
3. Let D_x be a distribution over integer solutions with expectation x.
4. Output a sample from D_x.

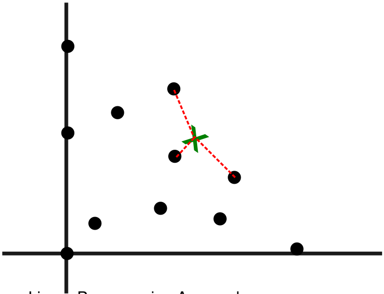


If scaled LP inside the convex hull of integer solutions, then algorithm is well-defined and MIDR over $\mathcal{R} = \left\{D_x : x \in \frac{1}{\alpha}P\right\}$.

Algorithm Outline

MIDR α-approximate Algorithm

1. Scale the feasible region P of the LP relaxation down by the integrality gap α.
2. Find the optimal solution x of the scaled LP.
3. Let D_x be a distribution over integer solutions with expectation x.
4. Output a sample from D_x.



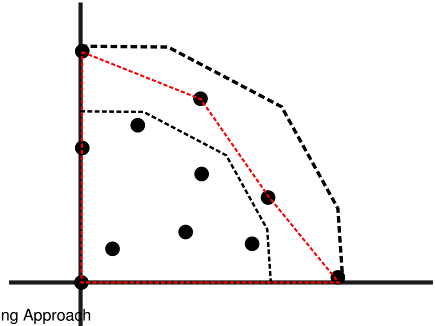
To be implementable in polynomial time, must show how to efficiently sample from D_x.

Scaling Lemma and Proof

Lemma

Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha}P \subseteq I$$



Scaling Lemma and Proof

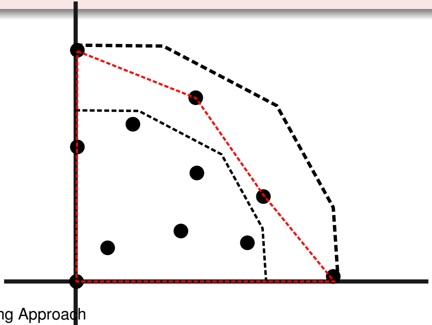
Lemma

Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha}P \subseteq I$$

Interpretation

Each feasible point of the scaled LP $\frac{1}{\alpha}P$ can be interpreted as a distribution over integer solutions.



Scaling Lemma and Proof

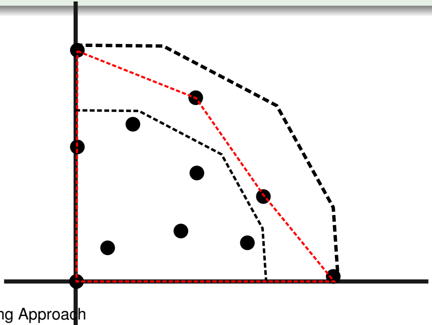
Lemma

Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha}P \subseteq I$$

The Separating Hyperplane Theorem

Let X and Y be two disjoint convex sets in euclidean space. There is a hyperplane h separating X and Y.



Scaling Lemma and Proof

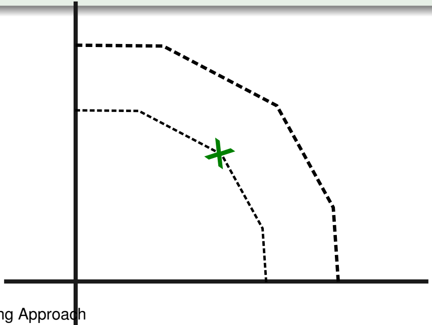
Lemma

Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha}P \subseteq I$$

The Separating Hyperplane Theorem

Let X and Y be two disjoint convex sets in euclidean space. There is a hyperplane h separating X and Y.



Scaling Lemma and Proof

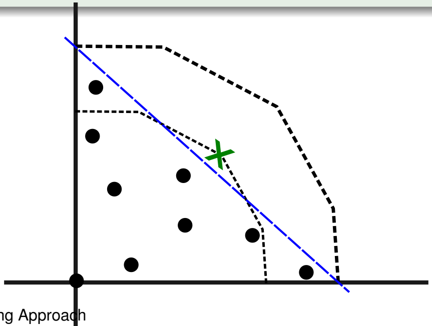
Lemma

Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha}P \subseteq I$$

The Separating Hyperplane Theorem

Let X and Y be two disjoint convex sets in euclidean space. There is a hyperplane h separating X and Y.



Scaling Lemma and Proof

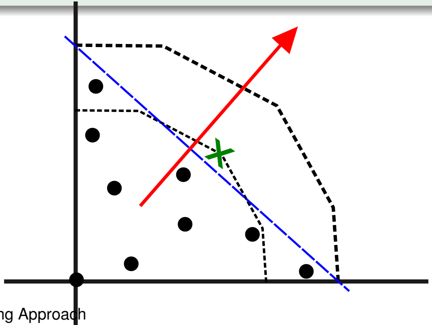
Lemma

Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha} P \subseteq I$$

The Separating Hyperplane Theorem

Let X and Y be two disjoint convex sets in euclidean space. There is a hyperplane h separating X and Y.



Scaling Lemma and Proof

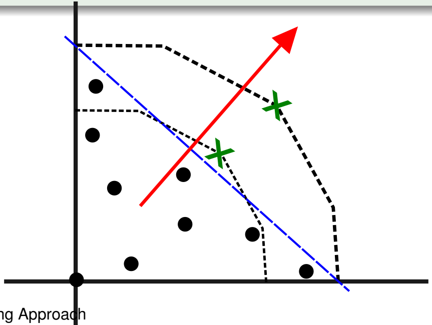
Lemma

Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha}P \subseteq I$$

The Separating Hyperplane Theorem

Let X and Y be two disjoint convex sets in euclidean space. There is a hyperplane h separating X and Y.



Scaling Lemma and Proof

Lemma

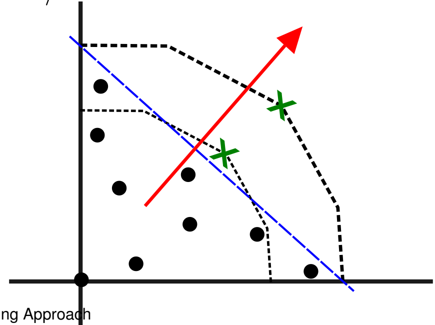
Let P be the polytope from the LP, with integrality gap α. Let I be the convex hull of its integer points.

$$\frac{1}{\alpha} P \subseteq I$$

But. . .

This argument breaks if normal to hyperplane points doesn't point "up"

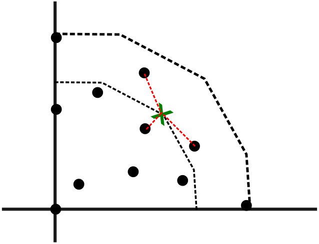
Can't happen since P/α is downwards closed



Sampling Lemma

Lemma

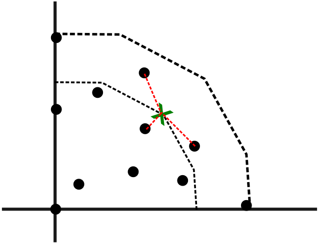
Assume black-box access to algorithm showing integrality gap α for the linear program P. For every point $x \in \frac{1}{\alpha}P$, we can efficiently construct a polynomial-sized-support distribution D_x on I with $\mathbf{E}_{y \sim D_x}[y] = x$.



Sampling Lemma

Lemma

Assume black-box access to algorithm showing integrality gap α for the linear program P. For every point $x \in \frac{1}{\alpha}P$, we can efficiently construct a polynomial-sized-support distribution D_x on I with $\mathbf{E}_{y \sim D_x}[y] = x$.



A distribution D_x with small support exists by Caratheodory's theorem

Caratheodory's Theorem

Let $X = \{x_1, \dots, x_k\} \subseteq \mathbb{R}^d$ and $y \in \mathbb{R}^d$. If $y \in convexhull(X)$ then there is $X' \subseteq X$ with $|X'| \leq d+1$ such that $y \in convexhull(X')$.

Intuition behind Sampling Lemma

Proof of Sampling Lemma

Applications of this framework

- The Lavi/Swamy framework establishes a tight connection between linear programming and mechanism design.
- Since LP is commonly used for the design of approximation algorithms, it is unsurprising that this framework has many applications :
    - Generalized assignment problem: 2
    - Combinatorial auctions with general valuations: $\sqrt{m}$
    - Multi-unit auctions: 2
    - $\ldots$