

Computationally Efficient Approximation Mechanisms by Ron Lavi

Capita Selecta *Algorithmic Game Theory*

Tomas Klos

February 22, 2008

'Approximation Mechanisms'?

February 22, 2008

Algorithmics Group

2

‘Approximation Mechanisms’?

- Design of Algorithms vs. Mechanisms
 - Similar aim: ‘implement desirable properties’
 - Which ones? e.g. \max welfare vs. \min makespan
 - Computational Efficiency

‘Approximation Mechanisms’?

- Design of Algorithms vs. Mechanisms
 - Similar aim: ‘implement desirable properties’
 - Which ones? e.g. \max welfare vs. \min makespan
 - Computational Efficiency
- Comp. Problems with Selfish Agents
 - Agents keep parameters of the problem secret
 - Merge CS and GT: combine good properties

‘Approximation Mechanisms’?

- Design of Algorithms vs. Mechanisms
 - Similar aim: ‘implement desirable properties’
 - Which ones? e.g. \max welfare vs. \min makespan
 - Computational Efficiency
- Comp. Problems with Selfish Agents
 - Agents keep parameters of the problem secret
 - Merge CS and GT: combine good properties:
 - So: induce them to reveal this ‘type’ truthfully
 - Then solve the resulting algorithmic problem

'Approximation Mechanisms'?

- Design of Algorithms vs. Mechanisms
 - Similar aim: 'implement desirable properties'
 - Which ones? e.g. \max welfare vs. \min makespan
 - Computational Efficiency
- Comp. Problems with Selfish Agents
 - Agents keep parameters of the problem secret
 - Merge CS and GT: combine good properties:
 - So: induce them to reveal this 'type' truthfully
 - Then solve the resulting algorithmic problem
- For which problems can we do this?

In this Chapter

Classical ...

- ... MD solutions computationally inefficient
- ... Algorithmic solutions game theoretically unsound

In this Chapter

Classical ...

- ... MD solutions computationally inefficient
- ... Algorithmic solutions game theoretically unsound

How bad is this clash, (when) can we resolve it?

In this Chapter

Classical ...

- ... MD solutions computationally inefficient
- ... Algorithmic solutions game theoretically unsound

How bad is this clash, (when) can we resolve it?

Related to dimensionality (of agents' valuation):

- Single-D
- Multi-D

In this Chapter

Classical ...

- ... MD solutions computationally inefficient
- ... Algorithmic solutions game theoretically unsound

How bad is this clash, (when) can we resolve it?

Related to dimensionality (of agents' valuation):

- Single-D: 'no (further) problem'
- Multi-D: hmmm ...

In this Chapter

Classical ...

- ... MD solutions computationally inefficient
- ... Algorithmic solutions game theoretically unsound

How bad is this clash, (when) can we resolve it?

Related to dimensionality (of agents' valuation):

- Single-D: 'no (further) problem' (Scheduling)
- Multi-D: hmmm ... (Combinatorial Auctions)

Chapter/Presentation Outline

Single dimensionality:

Multi dimensionality:

Chapter/Presentation Outline

Single dimensionality:

- Scheduling related machines $Q||C_{\max}$ (NP-hard)
- monotonicity condition imposed by truthfulness
- computationally efficient approximation mechanisms

Multi dimensionality:

Chapter/Presentation Outline

Single dimensionality:

- Scheduling related machines $Q||C_{\max}$ (NP-hard)
- monotonicity condition imposed by truthfulness
- computationally efficient approximation mechanisms

Multi dimensionality:

- Combinatorial Auctions (NP-hard, Chapter 11)
- bla

Chapter/Presentation Outline

Single dimensionality:

- Scheduling related machines $Q||C_{\max}$ (NP-hard)
- monotonicity condition imposed by truthfulness
- computationally efficient approximation mechanisms

Multi dimensionality:

- Combinatorial Auctions (NP-hard, Chapter 11)
- bla

Other solution concepts

Scheduling Related Machines

February 22, 2008

Algorithmics Group

5

Scheduling Related Machines

- CS Problem (NP-hard, PARTITION):
 - n jobs, m machines, each job j takes time p_j
 - machine i has speed s_i , needs p_j/s_i time for job j
 - work (load) for machine i is $w_i = \sum_{j \text{ assigned to } i} p_j$
 - minimize makespan (completion time): $\max_i w_i/s_i$

Scheduling Related Machines

- CS Problem (NP-hard, PARTITION):
 - n jobs, m machines, each job j takes time p_j
 - machine i has speed s_i , needs p_j/s_i time for job j
 - work (load) for machine i is $w_i = \sum_{j \text{ assigned to } i} p_j$
 - minimize makespan (completion time): $\max_i w_i/s_i$
- MD (=CS+GT) Problem:
 - machines are agents with (secret) type $t_i = 1/s_i$
 - mechanism may pay agents P to offset their costs
 - agents maximize profit = payment - cost = $P_i - w_i/s_i$

Approaches

- for CS problem
 - greedy LPT (2-approx) (Kleinberg & Tardos, Ch. 11)
 - PTAS (Hochbaum & Shmoys, 1988)

Approaches

- for CS problem
 - greedy LPT (2-approx) (Kleinberg & Tardos, Ch. 11)
 - PTAS (Hochbaum & Shmoys, 1988)
- for MD problem: can't use VCG
 - only for utilitarian objective (max total valuations)
 - requires optimal allocation

Approaches

- for CS problem
 - greedy LPT (2-approx) (Kleinberg & Tardos, Ch. 11)
 - PTAS (Hochbaum & Shmoys, 1988)
- for MD problem: can't use VCG
 - only for utilitarian objective (max total valuations)
 - requires optimal allocation
- induce truth-telling using payments
 - idea: if tt leads to increased load, compensate
 - **condition:** load curve may not increase

Monotonicity (Archer & Tardos 2001)

- agent i 's type $t_i = 1/s_i$, reports b_i
- type-vector is t , bid-vector is $b = (b_{-i}, b_i)$
- mechanism's output alg computes $o(b) \in O$
- outcome $o(b)$ assigns work $w_i(o(b))$ to i
- agent i has cost $\text{cost}_i(t_i, o(b)) = w_i/s_i = t_i w_i(o(b))$
- mechanism makes payment $P_i(b)$ to agent i
- agent i makes profit $\text{profit}_i(t_i, b) = P_i(b) - \text{cost}_i(t_i, o(b))$

Monotonicity (cont.)

- a strategy (reporting b_i) is *dominant* if
$$\text{profit}_i(t_i, (b_{-i}, b_i)) \geq \text{profit}_i(t_i, (b_{-i}, b'_i)) \quad \forall b_{-i}, b'_i$$

Monotonicity (cont.)

- a strategy (reporting b_i) is *dominant* if
$$\text{profit}_i(t_i, (b_{-i}, b_i)) \geq \text{profit}_i(t_i, (b_{-i}, b'_i)) \quad \forall b_{-i}, b'_i$$
- a mechanism is *truthful* if truth-telling (reporting t_i) is a dominant strategy for all i

Monotonicity (cont.)

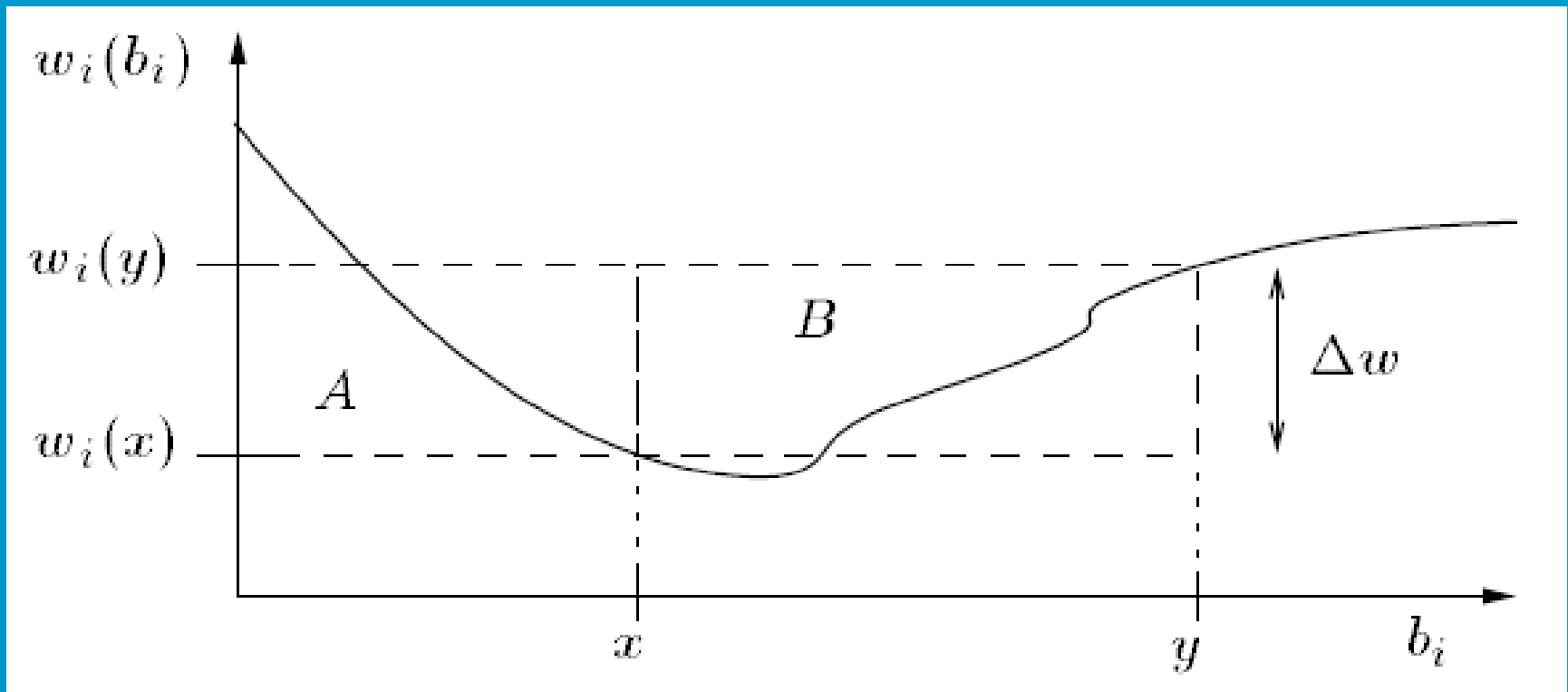
- a strategy (reporting b_i) is *dominant* if
$$\text{profit}_i(t_i, (b_{-i}, b_i)) \geq \text{profit}_i(t_i, (b_{-i}, b'_i)) \quad \forall b_{-i}, b'_i$$
- a mechanism is *truthful* if truth-telling (reporting t_i) is a dominant strategy for all i
- output function $o(b)$ *admits* a truthful payment scheme, if there exist payments P s.t. $M = (o, P)$ is truthful

Monotonicity (cont.)

- a strategy (reporting b_i) is *dominant* if
$$\text{profit}_i(t_i, (b_{-i}, b_i)) \geq \text{profit}_i(t_i, (b_{-i}, b'_i)) \quad \forall b_{-i}, b'_i$$
- a mechanism is *truthful* if truth-telling (reporting t_i) is a dominant strategy for all i
- output function $o(b)$ *admits* a truthful payment scheme, if there exist payments P s.t. $M = (o, P)$ is truthful
- if cost $\sim t_i w_i(o(b))$ (private cost/unit \times work assigned):
 - M is truthful $\leftrightarrow w'_i(b_i) \leq 0$, i.e. $o(b)$ admits a truthful payment scheme iff it is decreasing for all i
 - $P_i(b_{-i}, b_i) = b_i w_i(b_{-i}, b_i) + \int_{b_i}^{\infty} w_i(b_{-i}, u) du$

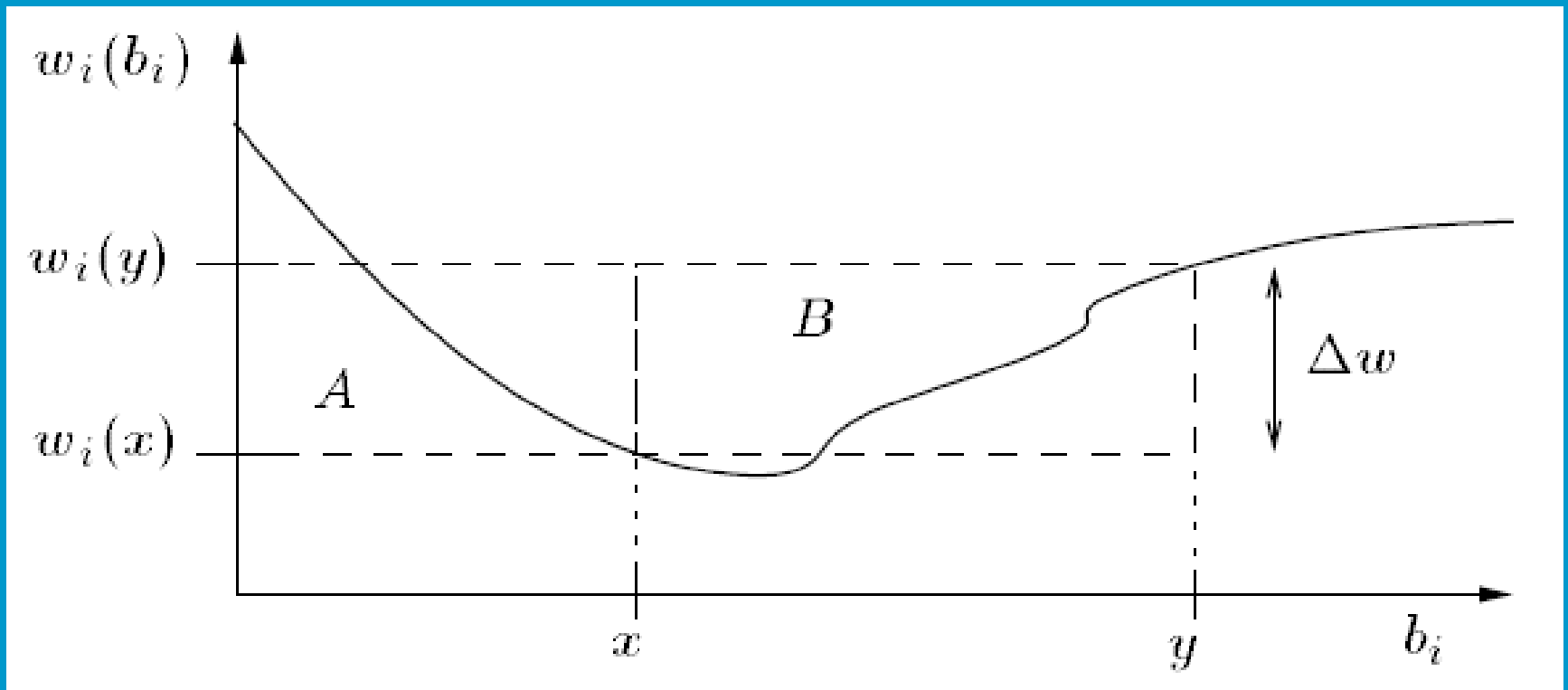
Monotonicity (cont.)

Work curve can not increase

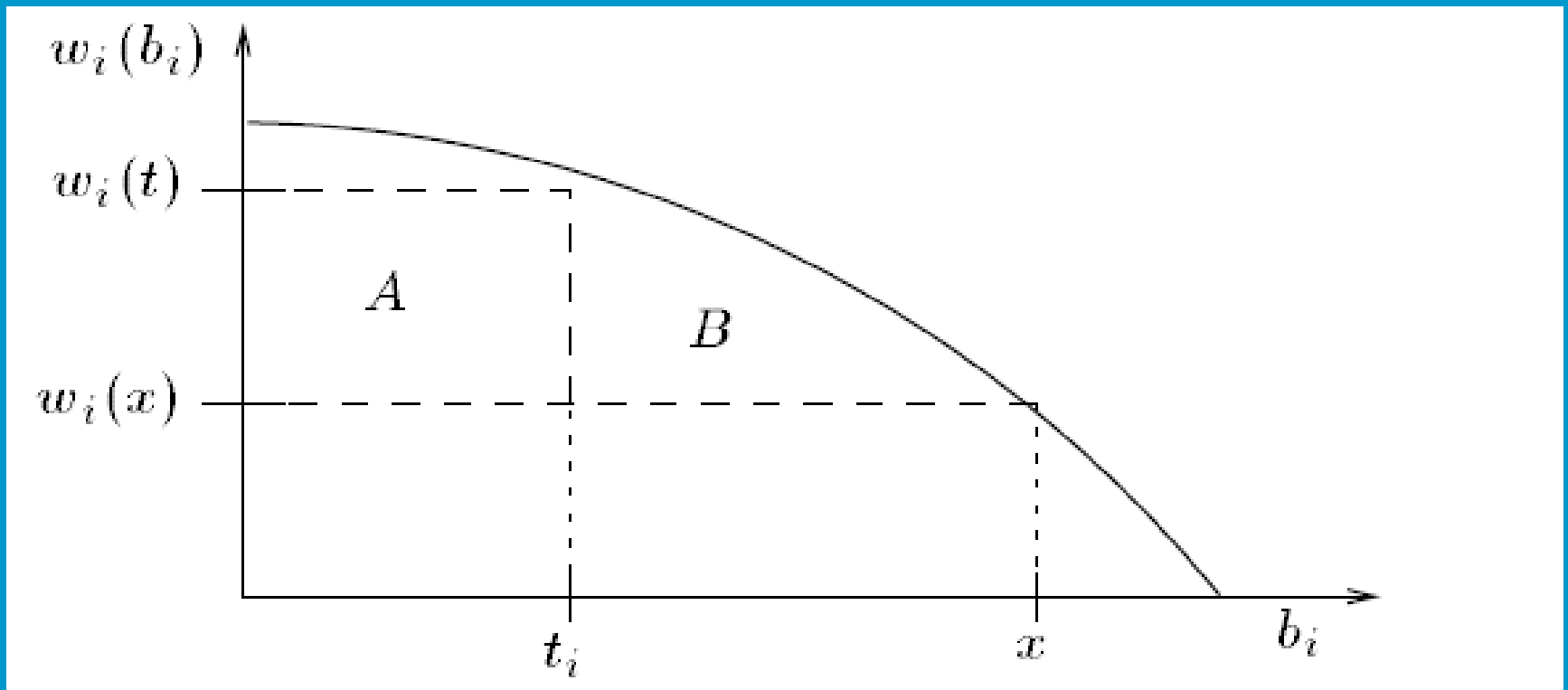


Monotonicity (cont.)

Work curve can not increase $A + B \leq P(y) \leq A$



Monotonicity (cont.)



bid $x > t_i$: reduces cost by A , payment by $A + B$

Output Algorithms

- GLB (LPT) and PTAS not monotone:
 - LPT: $p_1 = 2, p_2 = p_3 = 1 + \epsilon; s_1 > s_2$
 - PTAS: dyn. prog. + *rounding*

Output Algorithms

- GLB (LPT) and PTAS not monotone:
 - LPT: $p_1 = 2, p_2 = p_3 = 1 + \epsilon; s_1 > s_2$
 - PTAS: dyn. prog. + *rounding*
- algorithms with monotone output function

Output Algorithms

- GLB (LPT) and PTAS not monotone:
 - LPT: $p_1 = 2, p_2 = p_3 = 1 + \epsilon; s_1 > s_2$
 - PTAS: dyn. prog. + *rounding*
- algorithms with monotone output function
 - randomized (Archer & Tardos)
 - 3-approx
 - truthful-in-expectation

Output Algorithms

- GLB (LPT) and PTAS not monotone:
 - LPT: $p_1 = 2, p_2 = p_3 = 1 + \epsilon; s_1 > s_2$
 - PTAS: dyn. prog. + *rounding*
- algorithms with monotone output function
 - randomized (Archer & Tardos)
 - 3-approx
 - truthful-in-expectation
 - deterministic (Andelman, Azar & Sorani)
 - 5-approx
 - truthful

Output Algorithms

- GLB (LPT) and PTAS not monotone:
 - LPT: $p_1 = 2, p_2 = p_3 = 1 + \epsilon; s_1 > s_2$
 - PTAS: dyn. prog. + *rounding*
- algorithms with monotone output function
 - randomized (Archer & Tardos)
 - 3-approx, current best is 2-approx
 - truthful-in-expectation
 - deterministic (Andelman, Azar & Sorani)
 - 5-approx, current best is 3-approx
 - truthful

Randomized Monotone Algorithm

1. guess C_{\max}^* at T
2. lowerbound T in valid *fractional* assignment
3. greedily produce valid fractional assignment
4. randomized-round the fractional assignment

Randomized Monotone Algorithm

1. guess C_{\max}^* at T (target makespan)

- (problem is bin packing with uneven bins)
- create m bins sized T/b_i (max. load to finish by T)
- obs: $T \geq C_{\max}^* \leftrightarrow \exists$ assignment s.t. $\forall i, \sum_{j|i} p_j \leq \text{bin}_i$

Randomized Monotone Algorithm

1. guess C_{\max}^* at T (target makespan)

- (problem is bin packing with uneven bins)
- create m bins sized T/b_i (max. load to finish by T)
- obs: $T \geq C_{\max}^* \leftrightarrow \exists$ assignment s.t. $\forall i, \sum_{j|i} p_j \leq \text{bin}_i$

2. lowerbound T in valid *fractional* assignment

- allow *fractional* assignment (FA): partition each job
- *valid*(FA): (1) $\text{bin}_i \geq \sum_{j|i} p_j^i$ and (2) $\text{bin}_i \geq p_j \quad \forall i, j|i$
- smallest T s.t. a valid FA exists, lowerbounds C_{\max}^*

Randomized Monotone Algorithm

3. greedily produce valid fractional assignment

- order: $b_1 \leq b_2 \leq \dots \leq b_m$ and $p_1 \geq p_2 \geq \dots \geq p_n$
- greedily assign jobs to bins fractionally (GFA)
- valid(GFA)?

Randomized Monotone Algorithm

3. greedily produce valid fractional assignment

- order: $b_1 \leq b_2 \leq \dots \leq b_m$ and $p_1 \geq p_2 \geq \dots \geq p_n$
- greedily assign jobs to bins fractionally (GFA)
- valid(GFA)?
 - for each job j , $i(j)$ is largest i s.t. $\text{bin}_i \geq p_j$
 - valid(GFA) iff $\forall j, \sum_{k=1}^{i(j)} \text{bin}_k \geq \sum_{l=1}^j p_l$

Randomized Monotone Algorithm

3. greedily produce valid fractional assignment

- order: $b_1 \leq b_2 \leq \dots \leq b_m$ and $p_1 \geq p_2 \geq \dots \geq p_n$
- greedily assign jobs to bins fractionally (GFA)
- valid(GFA)?
 - for each job j , $i(j)$ is largest i s.t. $\text{bin}_i \geq p_j$
 - valid(GFA) iff $\forall j, \sum_{k=1}^{i(j)} \text{bin}_k \geq \sum_{l=1}^j p_l$
- if valid(GFA) and $i(j)$ is largest k s.t. $j|k$, then

Randomized Monotone Algorithm

3. greedily produce valid fractional assignment

- order: $b_1 \leq b_2 \leq \dots \leq b_m$ and $p_1 \geq p_2 \geq \dots \geq p_n$
- greedily assign jobs to bins fractionally (GFA)
- valid(GFA)?
 - for each job j , $i(j)$ is largest i s.t. $\text{bin}_i \geq p_j$
 - valid(GFA) iff $\forall j, \sum_{k=1}^{i(j)} \text{bin}_k \geq \sum_{l=1}^j p_l$
- if valid(GFA) and $i(j)$ is largest k s.t. $j|k$, then

$$T_{LB} = \max \left\{ b_i p_j, \frac{\sum_{l=1}^j p_l}{\sum_{k=1}^{i(j)} s_k} \right\}$$

Randomized Monotone Algorithm

3. greedily produce valid fractional assignment

- order: $b_1 \leq b_2 \leq \dots \leq b_m$ and $p_1 \geq p_2 \geq \dots \geq p_n$
- greedily assign jobs to bins fractionally (GFA)
- valid(GFA)?
 - for each job j , $i(j)$ is largest i s.t. $\text{bin}_i \geq p_j$
 - valid(GFA) iff $\forall j, \sum_{k=1}^{i(j)} \text{bin}_k \geq \sum_{l=1}^j p_l$
- if valid(GFA) and $i(j)$ is largest k s.t. $j|k$, then

$$T_{LB} = \min_i \max \left\{ b_i p_j, \frac{\sum_{l=1}^j p_l}{\sum_{k=1}^{i(j)} s_k} \right\}$$

Randomized Monotone Algorithm

3. greedily produce valid fractional assignment

- order: $b_1 \leq b_2 \leq \dots \leq b_m$ and $p_1 \geq p_2 \geq \dots \geq p_n$
- greedily assign jobs to bins fractionally (GFA)
- valid(GFA)?
 - for each job j , $i(j)$ is largest i s.t. $\text{bin}_i \geq p_j$
 - valid(GFA) iff $\forall j, \sum_{k=1}^{i(j)} \text{bin}_k \geq \sum_{l=1}^j p_l$
- if valid(GFA) and $i(j)$ is largest k s.t. $j|k$, then

$$T_{LB} = \max_j \min_i \max \left\{ b_i p_j, \frac{\sum_{l=1}^j p_l}{\sum_{k=1}^{i(j)} s_k} \right\}$$

Randomized Monotone Algorithm

4. randomized-round the fractional assignment

- size bins using T_{LB} : $\text{bin}_i = T_{LB}/b_i$
- assigning j to $\arg \max_i |j \text{ is assigned to } i| \cdot s_i$:
2-approx but not monotone

Randomized Monotone Algorithm

4. randomized-round the fractional assignment

- size bins using T_{LB} : $\text{bin}_i = T_{LB}/b_i$
- assigning j to $\arg \max_{i|j \text{ is assigned to } i} s_i$:
2-approx but not monotone
- try randomized algorithm
- *truthful-in-expectation*: t_i maximizes *expected* profit
- assign j to each machine i with probability proportional to fraction of j assigned to i

Randomized Monotone Algorithm

4. randomized-round the fractional assignment

- size bins using T_{LB} : $\text{bin}_i = T_{LB}/b_i$
- assigning j to $\arg \max_{i|j \text{ is assigned to } i} s_i$:
2-approx but not monotone
- try randomized algorithm
- *truthful-in-expectation*: t_i maximizes *expected* profit
- assign j to each machine i with probability proportional to fraction of j assigned to i
- 3-approximation: max 2 jobs extra for machine i

Multi-dimensional Valuations

Recap:

- problems where selfish agents hold the inputs
- AMD: efficiently computable truthful mechanisms
- not all algs lead to truthful (VCG) mechanisms (e.g. for NP-hard problems): VCG requires exact solution
- approx. algs not always applicable (esp. multi-D domains: very complex monotonicity cond.)

General Approach (Lavy & Swamy)

convert approx. algs into truthful mechanisms:

- (randomized mechanisms: truthful-in-expectation)
- relaxation (combinatorial, integer) problem to LP
- use that to design approx. alg (rounding/PD)
- convert that to randomized mechanism

General Approach (Lavy & Swamy)

convert approx. algs into truthful mechanisms:

- (randomized mechanisms: truthful-in-expectation)
- relaxation (combinatorial, integer) problem to LP
- use that to design approx. alg (rounding/PD)
- convert that to randomized mechanism

main theorem: (for packing domains) any approx. alg for the CS problem (that bounds the *integrality gap*) can be used to obtain a randomized, t-i-e mechanism for the corresponding MD problem, with approx. guarantee matching that of the approx. alg

MD Preliminaries

- n agents, set A of possible outcomes
- agents have type $v_i : A \mapsto \mathbb{R}_{\geq 0}$, $v_i \in V_i$
- $V = V_1 \times \dots \times V_n$ is space of all players' types, $v = (v_1, \dots, v_n)$, \bar{v}_i is i 's true type
- e.g. in CA: $A = \{(S_1, \dots, S_n) \text{ with } S_i \cap S_j = \emptyset\}$
- $M = (f, p)$ with $f : V \mapsto A$ and $p_i : V \mapsto \mathbb{R}$ for each i
- reports v , M computes $f(v)$ and charges $p_i(v)$ to i
- *randomized* M^R flips coins to determine $f(v)$ and $p_i(v)$
- t-i-e: reporting \bar{v}_i maximizes *expected* profit

Randomized Mechanism

- randomized alg: 'prob. dist. over deterministic algs'
- randomized mechanism: similar
- idea: for M^R we want, define M^D giving dist's
- we construct that M^D from solution of LP relaxation

Randomized Mechanism

- randomized alg: ‘prob. dist. over deterministic algs’
- randomized mechanism: similar
- idea: for M^R we want, define M^D giving dist’s
- we construct that M^D from solution of LP relaxation
- for A of M^R , define unit simplex for M^D :
$$\Delta_A = \{\lambda \in \mathbb{R}^{|A|} : \sum_{a \in A} \lambda_a = 1, \lambda_a \geq 0 \forall a\}$$
- e.g. for $A = \{a_1, a_2, a_3\}$, Δ_A is triangle $a_1 - a_2 - a_3$

Randomized Mechanism

- randomized alg: ‘prob. dist. over deterministic algs’
- randomized mechanism: similar
- idea: for M^R we want, define M^D giving dist’s
- we construct that M^D from solution of LP relaxation
- for A of M^R , define unit simplex for M^D :
$$\Delta_A = \{\lambda \in \mathbb{R}^{|A|} : \sum_{a \in A} \lambda_a = 1, \lambda_a \geq 0 \forall a\}$$
- e.g. for $A = \{a_1, a_2, a_3\}$, Δ_A is triangle $a_1 - a_2 - a_3$
- for any $M^R = (f^R, p^R)$, define $M^D = (\Delta_A, E(p^R))$
- extend domain V_i : $v_i(\{\lambda_a\}_{a \in A}) = \sum_{a \in A} \lambda_a v_i(a)$

Randomized Mechanism

- M^D is *deterministic support mechanism (DSM)* of M^R
- $f^R(v) \in A$ is random variable with dist. $f^D(v)$
- M^R outputs a with prob. λ_a
- prices of M^R have to maintain individual rationality...

Randomized Mechanism

- M^D is *deterministic support mechanism (DSM)* of M^R
- $f^R(v) \in A$ is random variable with dist. $f^D(v)$
- M^R outputs a with prob. λ_a
- prices of M^R have to maintain individual rationality...

thm: given an ir, truthful DSM that computes α -approx and has only poly many $\lambda_a > 0$, one can obtain in poly-time an ir, t-i-e M^R that computes an α -approx to social welfare

Randomized Mechanism

- M^D is *deterministic support mechanism (DSM)* of M^R
- $f^R(v) \in A$ is random variable with dist. $f^D(v)$
- M^R outputs a with prob. λ_a
- prices of M^R have to maintain individual rationality...

thm: given an ir, truthful DSM that computes α -approx and has only poly many $\lambda_a > 0$, one can obtain in poly-time an ir, t-i-e M^R that computes an α -approx to social welfare

→ so how do we get M^D ?

Outline of Construction

applied to CA domain

- m items (Ω) allocated to n agents
- agents value bundles: $v_i(S)$ is i 's value for $S \subseteq \Omega$
- find (S_1, S_2, \dots, S_n) that maximizes $\sum_i v_i(S_i)$
- relaxation ($x_{i,S} \geq 0$, not $x_{i,S} \in \{0, 1\}$) is LP

Outline of Construction (cont.)

- fractional domain (F): optimally solvable in poly-time
- \rightarrow (truthful) VCG mechanism M^F , returns fractional feasible solution x^* and prices p^F .
- integrality gap $\alpha \geq 1$, social welfare $\geq \frac{x^*}{\alpha}$

Outline of Construction (cont.)

- fractional domain (F): optimally solvable in poly-time
- \rightarrow (truthful) VCG mechanism M^F , returns fractional feasible solution x^* and prices p^F .
- integrality gap $\alpha \geq 1$, social welfare $\geq \frac{x^*}{\alpha}$
- poly-time decomposition of $\frac{x^*}{\alpha}$ into polynomially many integer solutions \mathcal{I} , i.e. values λ_l s.t.
 $frac{x^*}{\alpha} = \sum_{l \in \mathcal{I}} \lambda_l x^l$ where $\{x^l\}_{l \in \mathcal{I}}$ is the set of all integer solutions to LP, $\lambda_l \geq 0$, and $\sum \lambda_l = 1$
- converts truthful M^F into truthful DSM
 $M^D = (\{\lambda_l\}_{l \in \mathcal{I}}, \frac{p^F}{\alpha})$

Conclusions

- single-D (Archer & Tardos)
 - extra req: monotonicity
 - tweak the alg to make is satisfy monotonicity
 - at expense of approx. guarantee
- multi-D (Ravy & Swamy)
 - no tweaking necessary
 - general (packing): (MU)CA's, graph routing, MU auctions
 - approx. guarantee (α) maintained