Computationally Efficient Approximation Mechanisms

AGT – capítulo 12 (por Ron Lavi)

Samuel Plaça de Paula

Junho de 2013

Algoritmos vs. Mecanismos

Almejam implementar propriedades desejáveis

Algoritmos vs. Mecanismos

Almejam implementar propriedades desejáveis

Necessidades e limitações inspiradas em situações reais

Algoritmos vs. Mecanismos

Almejam implementar propriedades desejáveis

Necessidades e limitações inspiradas em situações reais

Mas...

Propriedades desejadas resultam diferentes em cada abordagem!

Algoritmos vs. Mecanismos

Abordagens **computacional** e de **teoria dos jogos** destacam **propriedades distintas**.

Algoritmos vs. Mecanismos

Abordagens **computacional** e de **teoria dos jogos** destacam **propriedades distintas**.

Consequência natural: tentar conciliar as duas visões, obtendo propriedades boas sob ambos os pontos de vista.

Algoritmos vs. Mecanismos

Abordagens **computacional** e de **teoria dos jogos** destacam **propriedades distintas**.

Consequência natural: tentar conciliar as duas visões, obtendo propriedades boas sob ambos os pontos de vista.

Grande desafio!

Algoritmos vs. Mecanismos

Caso frequente

Um único problema estudado por ambas as disciplinas.

Algoritmos vs. Mecanismos

Caso frequente

Um único problema estudado por ambas as disciplinas.

Bons mecanismos (em relação a incentivos): computacionalmente **ineficientes**

Algoritmos vs. Mecanismos

Caso frequente

Um único problema estudado por ambas as disciplinas.

Bons mecanismos (em relação a incentivos): computacionalmente **ineficientes**

Bons algoritmos conhecidos: **não dão** bons mecanismos

Exemplo de conflito

Exemplo: escalonamento de tarefas

Objetivos geralmente diferem:
Makespan vs. bem-estar social.

Exemplo de conflito

Exemplo: escalonamento de tarefas

Objetivos geralmente diferem:
Makespan vs. bem-estar social.

Ou seja, geralmente um "não resolve" o outro.

Exemplo de conflito

Exemplo: escalonamento de tarefas

Objetivos geralmente diferem:
Makespan vs. bem-estar social.

Ou seja, geralmente um "não resolve" o outro.

Além disso, algoritmos conhecidos não dão bons mecanismos.

Conciliação?

Esse conflito entre abordagens é, em algum sentido, fundamental?

Ou será que conseguimos conciliar os dois mundos com técnicas diferentes?

Conciliação?

Esse conflito entre abordagens é, em algum sentido, fundamental?

Ou será que conseguimos conciliar os dois mundos com técnicas diferentes?

Depende fortemente da **dimensionalidade** do problema.

Voltando ao capítulo 9...

- n jogadores

Voltando ao capítulo 9...

- n jogadores
- Conjunto A de opções

Voltando ao capítulo 9...

- n jogadores
- Conjunto A de opções
- Cada jogador $i \in [n]$ tem avaliação $v_i \colon A \to \mathbb{R}$.

Voltando ao capítulo 9...

- n jogadores
- Conjunto A de opções
- Cada jogador $i \in [n]$ tem avaliação $v_i \colon A \to \mathbb{R}$.

Cada v_i pertence a um conjunto de valorações possíveis $V_i \subseteq \mathbb{R}^A$.

V_i é o que chamamos de **domínio (de preferências)** de i.

Voltando ao capítulo 9...

- n jogadores
- Conjunto A de opções
- Cada jogador $i \in [n]$ tem avaliação $v_i \colon A \to \mathbb{R}$.

Cada v_i pertence a um conjunto de valorações possíveis $V_i \subseteq \mathbb{R}^A$.

V_i é o que chamamos de **domínio (de preferências)** de i.

Cada $v_i' \in V_i$ é um possível **tipo** do jogador i.

Mecanismo:

Mecanismo:

- Função de escolha social $f : V_1 \times \cdots \times V_n \to A$;

Mecanismo:

- Função de escolha social $f : V_1 \times \cdots \times V_n \to A$;
- Vetor de funções de pagamento $(p_1, \ldots, p_n)$, onde $p_i : V_1 \times \cdots \times V_n \to \mathbb{R}$ é a função de pagamento do jogador i.

Mecanismo:

- Função de escolha social $f : V_1 \times \cdots \times V_n \to A$;
- Vetor de funções de pagamento $(p_1, \ldots, p_n)$, onde $p_i : V_1 \times \cdots \times V_n \to \mathbb{R}$ é a função de pagamento do jogador i.

A utilidade do jogador i para $v \in V_1 \times \cdots \times V_n$ é

$$u_i(v) = v_i(f(v)) - p_i(v).$$

Lembrando...

Definição

Um mecanismo $(f, p_1, \ldots, p_n)$ *é* **incentivo-compatível** *se,*

Lembrando...

Definição

Um mecanismo $(f, p_1, \ldots, p_n)$ *é* **incentivo-compatível** *se,*

Para todo $i \in [n]$,

Lembrando...

Definição

Um mecanismo $(f, p_1, \ldots, p_n)$ *é* **incentivo-compatível** *se,*

Para todo $i \in [n]$,
Todo $v \in V_1 \times \cdots \times V_n$, *e*

Lembrando...

Definição

Um mecanismo $(f, p_1, \ldots, p_n)$ *é* **incentivo-compatível** *se,*

Para todo $i \in [n]$,
Todo $v \in V_1 \times \cdots \times V_n$, *e*
Todo $v_i' \in V_i$,

Lembrando...

Definição

Um mecanismo $(f, p_1, \ldots, p_n)$ *é* **incentivo-compatível** *se,*

Para todo $i \in [n]$,
Todo $v \in V_1 \times \cdots \times V_n$, *e*
Todo $v_i' \in V_i$,

Se $a = f(v_i, v_{-i})$ *e* $a' = f(v_i', v_{-i})$, *então*

Lembrando...

Definição

Um mecanismo $(f, p_1, \ldots, p_n)$ *é* **incentivo-compatível** *se,*

Para todo $i \in [n]$,
Todo $v \in V_1 \times \cdots \times V_n$, *e*
Todo $v_i' \in V_i$,

Se $a = f(v_i, v_{-i})$ *e* $a' = f(v_i', v_{-i})$, *então*

$$v_i(a) - p_i(v_i, v_{-i}) \geq v_i(a') - p_i(v_i', v_{-i}).$$

Funções de escolha social implementáveis

"Implementar" uma função de escolha social f significa encontrar um vetor de pagamentos $(p_1, \ldots, p_n)$ tal que

$$(f, p_1, \ldots, p_n)$$

é um mecanismo incentivo-compatível.

Funções de escolha social implementáveis

"Implementar" uma função de escolha social f significa encontrar um vetor de pagamentos $(p_1, \ldots, p_n)$ tal que

$$(f, p_1, \ldots, p_n)$$

é um mecanismo incentivo-compatível.

Que funções de escolha social são implementáveis?

Funções de escolha social implementáveis

Que funções de escolha social são implementáveis?

Dada uma f, como descobrir se existe um "bom mecanismo" que tenha f como função de escolha social?

Funções de escolha social implementáveis

Que funções de escolha social são implementáveis?

Dada uma f, como descobrir se existe um "bom mecanismo" que tenha f como função de escolha social?

Essa f pode ser a escolha dada por um algoritmo eficiente... como conciliar os dois mundos?

Funções de escolha social implementáveis

Que funções de escolha social são implementáveis?

Dada uma f, como descobrir se existe um "bom mecanismo" que tenha f como função de escolha social?

Essa f pode ser a escolha dada por um algoritmo eficiente... como conciliar os dois mundos?

E se f for dada por um **algoritmo de aproximação** para um problema difícil de otimização?

Caracterização de mecanismos incentivo-compatíveis

Teorema

Mecanismo $(f, p_1, \ldots, p_n)$ *é incentivo-compatível sse:*

Caracterização de mecanismos incentivo-compatíveis

Teorema

Mecanismo $(f, p_1, \ldots, p_n)$ *é incentivo-compatível sse:*
Para todo $i \in [n]$*, todo* v_{-i}*:*

Caracterização de mecanismos incentivo-compatíveis

Teorema

Mecanismo $(f, p_1, \ldots, p_n)$ *é incentivo-compatível sse:*
Para todo $i \in [n]$, *todo* v_{-i}*:*

i. *Para cada* $a \in A$ *existe* $p_a \in \mathbb{R}$ *tal que, se* $f(v_i, v_{-i}) = a$, *então* $p_i(v_i, v_{-i}) = p_a$.

Caracterização de mecanismos incentivo-compatíveis

Teorema

Mecanismo $(f, p_1, \ldots, p_n)$ *é incentivo-compatível sse:*
Para todo $i \in [n]$, *todo* v_{-i}*:*

i. *Para cada* $a \in A$ *existe* $p_a \in \mathbb{R}$ *tal que, se* $f(v_i, v_{-i}) = a$, *então* $p_i(v_i, v_{-i}) = p_a$.

ii. $f(v_i, v_{-i}) \in \arg\max\{v_i(a) - p_a : a \in \mathrm{Im} f(\cdot, v_{-i})\}$.

Caracterização de mecanismos incentivo-compatíveis

Ou seja, um mecanismo é incentivo-compatível se, e só se,

Caracterização de mecanismos incentivo-compatíveis

Ou seja, um mecanismo é incentivo-compatível se, e só se,

Para todo i, fixando-se v_{-i},

Caracterização de mecanismos incentivo-compatíveis

Ou seja, um mecanismo é incentivo-compatível se, e só se,

Para todo i, fixando-se v_{-i}, o pagamento de i só depende do resultado $a = f(v_i, v_{-i})$, e o mecanismo otimiza a utilidade de i.

Caracterização de mecanismos incentivo-compatíveis

Ou seja, um mecanismo é incentivo-compatível se, e só se,

Para todo i, fixando-se v_{-i}, o pagamento de i só depende do resultado $a = f(v_i, v_{-i})$, e o mecanismo otimiza a utilidade de i.

E quanto a encontrar uma f **implementável**?

Monotonicidade fraca

Definição

Uma função de escolha social f satisfaz **monotonicidade fraca** *(WMON) se, para todo i, e para todo v_{-i},*

Monotonicidade fraca

Definição

Uma função de escolha social f satisfaz **monotonicidade fraca** *(WMON) se, para todo i, e para todo v_{-i},*

Se $f(v_i, v_{-i}) = a \neq b = f(v'_i, v_{-i})$

Monotonicidade fraca

Definição

Uma função de escolha social f satisfaz **monotonicidade fraca** *(WMON) se, para todo i, e para todo v_{-i},*

Se $f(v_i, v_{-i}) = a \neq b = f(v'_i, v_{-i})$ *, então*

$$v_i(a) - v_i(b) \geq v'_i(a) - v'_i(b).$$

Monotonicidade fraca

Definição

Uma função de escolha social f satisfaz **monotonicidade fraca** *(WMON) se, para todo i, e para todo v_{-i},*

Se $f(v_i, v_{-i}) = a \neq b = f(v'_i, v_{-i})$, então

$$v_i(a) - v_i(b) \geq v'_i(a) - v'_i(b).$$

Isto é, se i muda de ideia e isso muda o resultado de f, então esse i não piorou de valor com o novo resultado.

Caracterização parcial de funções implementáveis

Teorema

Se $(f, p_1, \ldots, p_n)$ é mecanismo incentivo-compatível, então f satisfaz WMON.

Caracterização parcial de funções implementáveis

Teorema

Se $(f, p_1, \ldots, p_n)$ é mecanismo incentivo-compatível, então f satisfaz WMON.

Se todos os domínios de preferências V_i são convexos então, se f satisfaz WMON, existem preços $(p_1, \ldots, p_n)$ tais que $(f, p_1, \ldots, p_n)$ é mecanismo incentivo-compatível.

Caracterização parcial de funções implementáveis

Teorema

Se $(f, p_1, \ldots, p_n)$ é mecanismo incentivo-compatível, então f satisfaz WMON.

Se todos os domínios de preferências V_i são convexos então, se f satisfaz WMON, existem preços $(p_1, \ldots, p_n)$ tais que $(f, p_1, \ldots, p_n)$ é mecanismo incentivo-compatível.

"Quase" uma caracterização!

Caracterização parcial – problemas

Caracterização parcial – problemas

- O "quase" é importante: em geral, WMON não é suficiente.

Caracterização parcial – problemas

- O "quase" é importante: em geral, WMON não é suficiente.
- Que funções satisfazem WMON?

Caracterização parcial – problemas

- O "quase" é importante: em geral, WMON não é suficiente.
- Que funções satisfazem WMON?

  É uma condição bem local: para cada i, e cada v_{-i}.
  Existe caracterização global?

Caracterização parcial – problemas

- O "quase" é importante: em geral, WMON não é suficiente.
- Que funções satisfazem WMON?

  É uma condição bem local: para cada i, e cada v_{-i}.
  Existe caracterização global?

  Sim, para V_i com dimensionalidades extremas.

Escalonando máquinas relacionadas

- n tarefas para m máquinas. Os jogadores são as máquinas!

Escalonando máquinas relacionadas

- n tarefas para m máquinas. Os jogadores são as máquinas!
- Tarefa j consome $w_j \geq 0$ unidades de tempo, máquina i tem velocidade $s_i > 0$.

Escalonando máquinas relacionadas

- n tarefas para m máquinas. Os jogadores são as máquinas!
- Tarefa j consome $w_j \geq 0$ unidades de tempo, máquina i tem velocidade $s_i > 0$.
- Máquina i processa tarefa j em tempo $\frac{w_j}{s_i}$.

Escalonando máquinas relacionadas

- n tarefas para m máquinas. Os jogadores são as máquinas!
- Tarefa j consome $w_j \geq 0$ unidades de tempo, máquina i tem velocidade $s_i > 0$.
- Máquina i processa tarefa j em tempo $\frac{w_j}{s_i}$.
- Uma opção $a \in A$ é uma atribuição $a\colon [n] \to [m]$ das tarefas.

Escalonando máquinas relacionadas

Para máquina $i \in [m]$, escalonamento $a \in A$, **carga** de i é

$$l_i(a) = \sum_{j\colon a(j)=i} w_j,$$

Escalonando máquinas relacionadas

Para máquina $i \in [m]$, escalonamento $a \in A$, **carga** de i é

$$l_i(a) = \sum_{j\colon a(j)=i} w_j,$$

de modo que o valor da alternativa a, para i, é

$$v_i(a) = -\frac{l_i(a)}{s_i}.$$

Escalonando máquinas relacionadas

Note que a **valoração v_i é totalmente dada pela velocidade s_i** da máquina.

Escalonando máquinas relacionadas

Note que a **valoração v_i é totalmente dada pela velocidade s_i** da máquina.

Isso significa que o domínio de preferências V_i é simplesmente o conjunto de possíveis velocidades de i: **o domínio é unidimensional**.

Unidimensionalidade, linearidade

Definição

O domínio V_i do jogador i é **unidimensional** *e* **linear** *se existem constantes reais não negativas (as "cargas") $\{q_{i,a}\}_{a \in A}$ tais que, para cada $v_i \in V_i$, existe uma constante $c \in \mathbb{R}_-$ (o "custo") tal que $v_i(a) = q_{i,a} \cdot c$.*

Unidimensionalidade, linearidade

Definição

O domínio V_i do jogador i é **unidimensional** *e* **linear** *se existem constantes reais não negativas (as "cargas") $\{q_{i,a}\}_{a \in A}$ tais que, para cada $v_i \in V_i$, existe uma constante $c \in \mathbb{R}_-$ (o "custo") tal que $v_i(a) = q_{i,a} \cdot c$.*

No caso, para cada $i \in [m]$,

- Para cada $a \in A$, $q_{i,a}$ é a carga $l_i(a)$ (constante),
- $c = -\frac{1}{s_i}$.

Unidimensionalidade, linearidade

Definição

O domínio V_i do jogador i é **unidimensional** *e* **linear** *se existem constantes reais não negativas (as "cargas") $\{q_{i,a}\}_{a \in A}$ tais que, para cada $v_i \in V_i$, existe uma constante $c \in \mathbb{R}_-$ (o "custo") tal que $v_i(a) = q_{i,a} \cdot c$.*

No caso, para cada $i \in [m]$,

- Para cada $a \in A$, $q_{i,a}$ é a carga $l_i(a)$ (constante),
- $c = -\frac{1}{s_i}$.

O **tipo** do jogador i é simplesmente essa constante c.

Implementando uma função de escolha social

Desejamos, neste cenário, escolher uma função de escolha social para (tentar) implementar.

Implementando uma função de escolha social

Desejamos, neste cenário, escolher uma função de escolha social para (tentar) implementar.

Objetivo: minimizar *maskespan*. Isto é, queremos

$$\min_{a \in A} \max_{i \in [m]} \frac{l_i(a)}{s_i}.$$

Implementando uma função de escolha social

Desejamos, neste cenário, escolher uma função de escolha social para (tentar) implementar.

Objetivo: minimizar *maskespan*. Isto é, queremos

$$\min_{a \in A} \max_{i \in [m]} \frac{l_i(a)}{s_i}.$$

Problema 1: Não dá para implementar via mecanismo VCG, que é o que conhecemos.

Implementando uma função de escolha social

Desejamos, neste cenário, escolher uma função de escolha social para (tentar) implementar.

Objetivo: minimizar *maskespan*. Isto é, queremos

$$\min_{a \in A} \max_{i \in [m]} \frac{l_i(a)}{s_i}.$$

Problema 1: Não dá para implementar via mecanismo VCG, que é o que conhecemos.

Problema 2: Este problema é NP-difícil.

Alternativas...

Teremos que escolher uma função de escolha social que sejamos capazes de calcular de maneira eficiente.

Ou seja, iremos tentar implementar uma função de escolha social dada por um algoritmo de aproximação.

Alternativas...

Teremos que escolher uma função de escolha social que sejamos capazes de calcular de maneira eficiente.

Ou seja, iremos tentar implementar uma função de escolha social dada por um algoritmo de aproximação.

Que tipos de funções conseguimos implementar?

Alternativas...

Teremos que escolher uma função de escolha social que sejamos capazes de calcular de maneira eficiente.

Ou seja, iremos tentar implementar uma função de escolha social dada por um algoritmo de aproximação.

Que tipos de funções conseguimos implementar?

Como nossos domínios são convexos, sabemos que precisamos de uma função (algoritmo) **que satisfaça WMON**.

Satisfazendo WMON

Felizmente, pode ser bastante simples para domínios unidimensionais.

Satisfazendo WMON

Felizmente, pode ser bastante simples para domínios unidimensionais.

Fixe custos c_{-i} declarados pelos outros jogadores.
Algoritmo para domínios unidimensionais e lineares determina a carga $q_i(c)$ do jogador i como função de seu custo declarado c.

Satisfazendo WMON

Felizmente, pode ser bastante simples para domínios unidimensionais.

Fixe custos c_{-i} declarados pelos outros jogadores.
Algoritmo para domínios unidimensionais e lineares determina a carga $q_i(c)$ do jogador i como função de seu custo declarado c.

Tome dois possíveis tipos (custos) c e c', e suponha $c' > c$.

Satisfazendo WMON

Felizmente, pode ser bastante simples para domínios unidimensionais.

Fixe custos c_{-i} declarados pelos outros jogadores.
Algoritmo para domínios unidimensionais e lineares determina a carga $q_i(c)$ do jogador i como função de seu custo declarado c.

Tome dois possíveis tipos (custos) c e c', e suponha $c' > c$.
WMON se reduz a

$$-q_i(c')(c' - c) \geq -q_i(c)(c' - c),$$

que vale se, e só se, $q_i(c') \leq q_i(c)$.

Satisfazendo WMON

WMON: fixando c_{-i}, custo q_i (como função do custo declarado de i) tem que satisfazer:

Se $c' > c$, então $q_i(c') \leq q_i(c)$.

Satisfazendo WMON

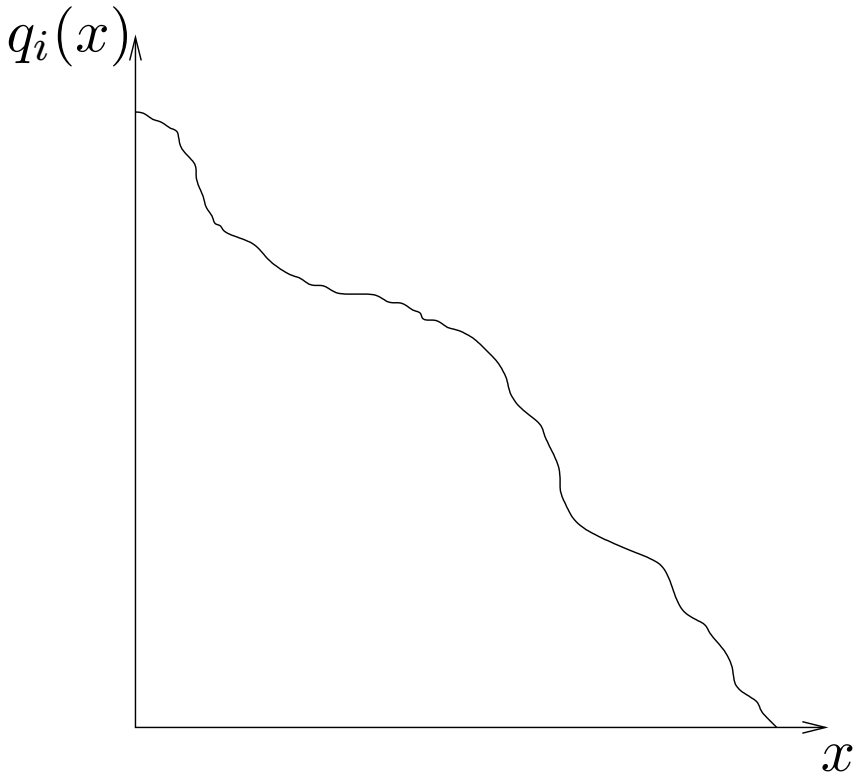
WMON: fixando c_{-i}, custo q_i (como função do custo declarado de i) tem que satisfazer:

Se $c' > c$, então $q_i(c') \leq q_i(c)$.

Ou seja, q_i tem que ser monotonicamente não crescente.

Monotonicidade

Ou seja, q_i tem que ser monotonicamente não crescente.



Pagamentos

Conhecemos a condição sobre nossa função de escolha social.

Pagamentos

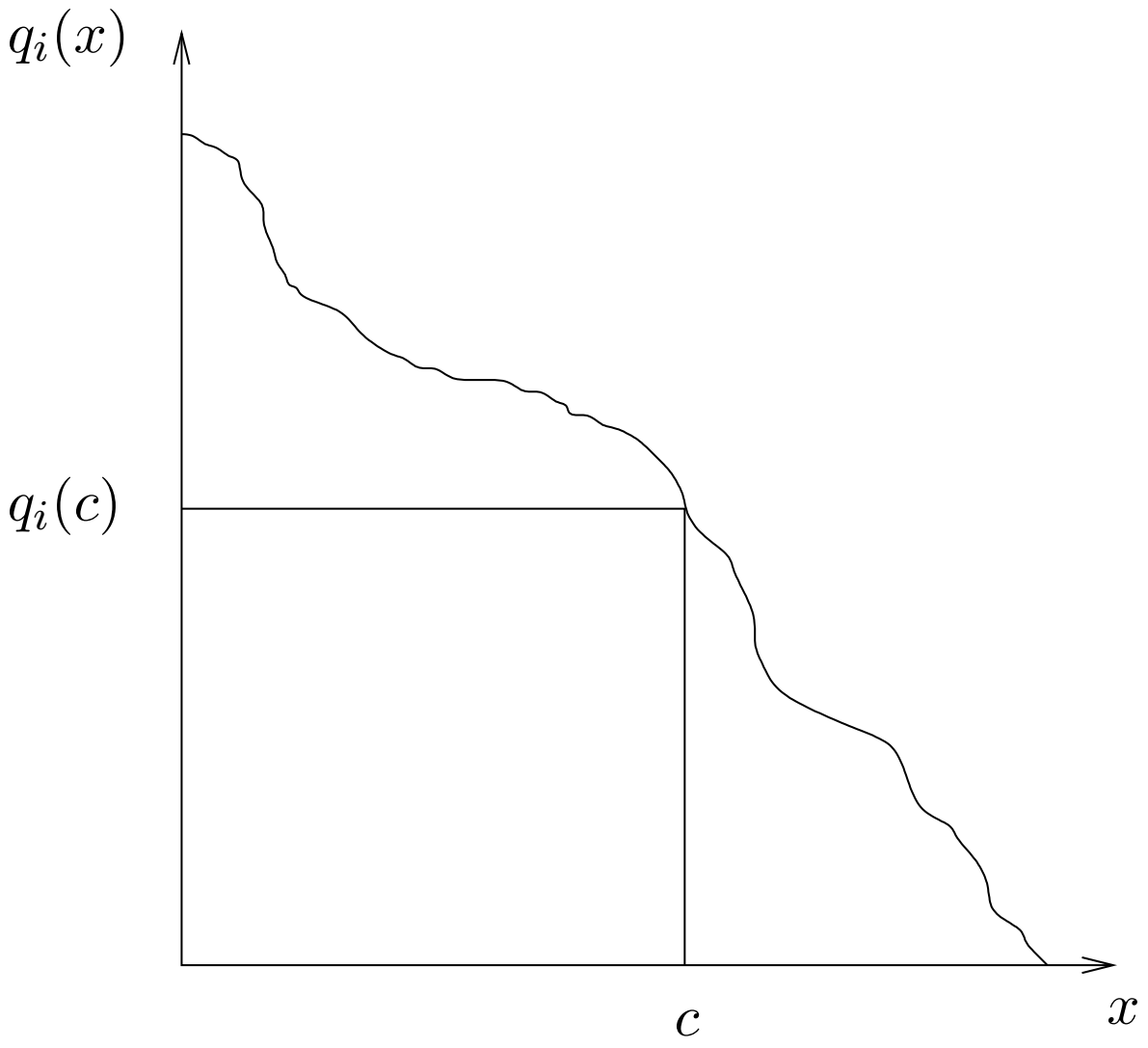
Conhecemos a condição sobre nossa função de escolha social.

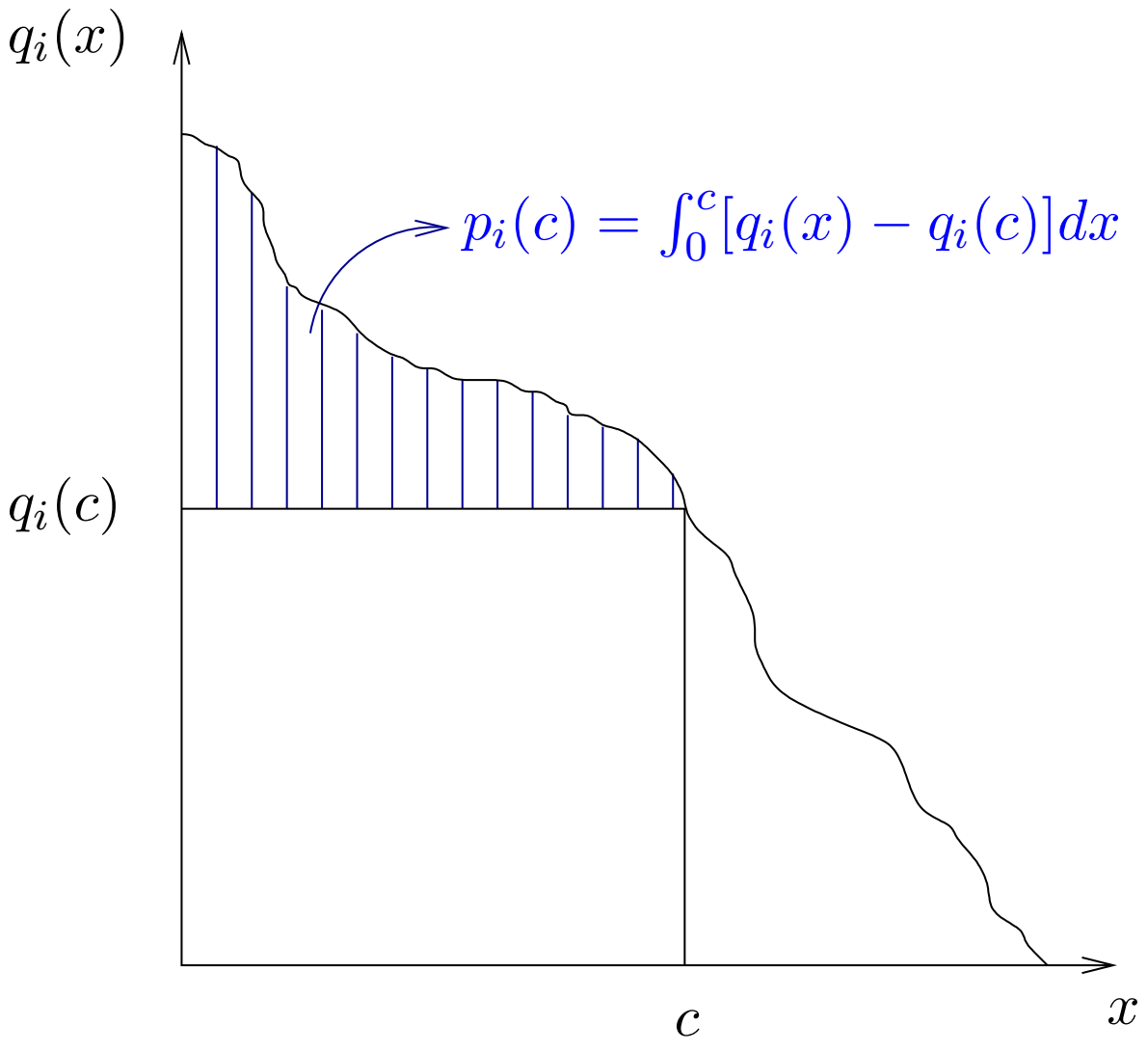
E os **pagamentos**?

Pagamentos

Conhecemos a condição sobre nossa função de escolha social.

E os **pagamentos**?
Queremos pagamentos que incentivem os jogadores a declarar seu tipo verdadeiro.


$q_i(x)$
$q_i(c)$
c
x


$q_i(x)$
$p_i(c) = \int_0^c [q_i(x) - q_i(c)]dx$
$q_i(c)$
c
x

Preços escolhidos

Os preços escolhidos são

$$p_i(c) = \int_0^c [q_i(x) - q_i(c)]dx.$$

Preços escolhidos

Os preços escolhidos são

$$p_i(c) = \int_0^c [q_i(x) - q_i(c)]dx.$$

Note que, fixando os custos c_{-i} declarados pelos jogadores, o custo do escalonamento, para o jogador i, é

$$-cq_i(c_d) - p_i(c_d),$$

onde c é o custo verdadeiro (tipo) de i e c_d é o custo declarado por i.

Preços escolhidos

Os preços escolhidos são

$$p_i(c) = \int_0^c [q_i(x) - q_i(c)]dx.$$

Note que, fixando os custos c_{-i} declarados pelos jogadores, o custo do escalonamento, para o jogador i, é

$$-cq_i(c_d) - p_i(c_d),$$

onde c é o custo verdadeiro (tipo) de i e c_d é o custo declarado por i.

Estes preços dão um mecanismo incentivo-compatível?

Preços

Estes preços dão um mecanismo incentivo-compatível?

Preços

Estes preços dão um mecanismo incentivo-compatível?

Sim! Observe o gráfico.

Preços

Estes preços dão um mecanismo incentivo-compatível?

Sim! Observe o gráfico.

No entanto, estes preços ainda não nos dão um mecanismo individualmente racional.
Precisamos tornar a utilidade não-negativa.

Preços

Estes preços dão um mecanismo incentivo-compatível?

Sim! Observe o gráfico.

No entanto, estes preços ainda não nos dão um mecanismo individualmente racional.
Precisamos tornar a utilidade não-negativa.

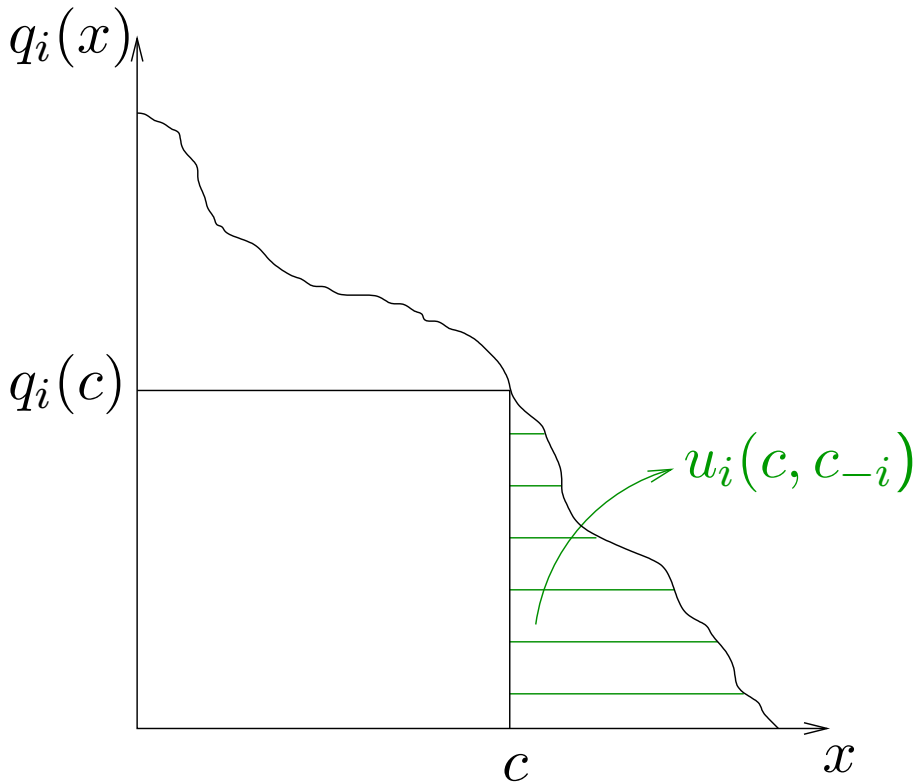
Novo preço:

$$p_i(c) = \int_0^c [q_i(x) - q_i(c)]dx - \int_0^\infty q_i(x)dx.$$

Utilidade com novo preço

Com estes preços, utilidade de i fica não negativa.
Se i declara seu tipo verdadeiro, então

$$u_i(c, c_{-i}) = \int_c^{\infty} q_i(x)dx$$



Portanto...

Teorema

Um algoritmo para domínios unidimensionais lineares é implementável se, e só se, suas funções de carga são não crescentes.

Portanto...

Teorema

Um algoritmo para domínios unidimensionais lineares é implementável se, e só se, suas funções de carga são não crescentes.

Além disso, se este é o caso, então cobrar de cada jogador i o pagamento

$$p_i(c) = \int_0^c [q_i(x) - q_i(c)]dx - \int_0^\infty q_i(x)dx$$

resulta num mecanismo incentivo-compatível individualmente racional.

E agora?

Conseguimos caracterizar os algoritmos que são implementáveis por bons mecanismos para o caso de domínios unidimensionais lineares.

E agora?

Conseguimos caracterizar os algoritmos que são implementáveis por bons mecanismos para o caso de domínios unidimensionais lineares.

Agora, iremos voltar para o problema específico do escalonamento.

E agora?

Conseguimos caracterizar os algoritmos que são implementáveis por bons mecanismos para o caso de domínios unidimensionais lineares.

Agora, iremos voltar para o problema específico do escalonamento. Sabemos tratar-se de um problema NP-difícil.

E agora?

Conseguimos caracterizar os algoritmos que são implementáveis por bons mecanismos para o caso de domínios unidimensionais lineares.

Agora, iremos voltar para o problema específico do escalonamento. Sabemos tratar-se de um problema NP-difícil.

Queremos mecanismos eficientemente computáveis. Para tanto, pelo que já vimos, queremos um **algoritmo de aproximação monotônico**.

Estimando makespan ótimo

Considere que:

Estimando makespan ótimo

Considere que:

- Máquinas ordenadas de modo que

$$s_1 \geq s_2 \geq \cdots \geq s_m,$$

Estimando makespan ótimo

Considere que:

- Máquinas ordenadas de modo que

$$s_1 \geq s_2 \geq \cdots \geq s_m,$$

- Tarefas ordenadas de modo que

$$w_1 \geq w_2 \geq \cdots \geq w_n.$$

Estimando makespan ótimo

Considere que:

- Máquinas ordenadas de modo que

$$s_1 \geq s_2 \geq \cdots \geq s_m,$$

- Tarefas ordenadas de modo que

$$w_1 \geq w_2 \geq \cdots \geq w_n.$$

Para construir o algoritmo, vamos primeiro tentar estimar o valor do makespan ótimo de uma instância.

Estimando makespan ótimo

Fixe uma tarefa j, e um possível valor de makespan T.

Estimando makespan ótimo

Fixe uma tarefa j, e um possível valor de makespan T.

Se escalonamento a possui makespan $\leq T$, então a deve designar cada tarefa em $\{1, \ldots, j\}$ a uma máquina i tal que $T \geq w_j/s_i$.

Estimando makespan ótimo

Fixe uma tarefa j, e um possível valor de makespan T.

Se escalonamento a possui makespan $\leq T$, então a deve designar cada tarefa em $\{1, \ldots, j\}$ a uma máquina i tal que $T \geq w_j/s_i$.

Defina $i(j, T) = \max\{i : T \geq w_j/s_i\}$, isto é, a máquina mais lenta que não viola a restrição para j (a tarefa mais leve).

Estimando makespan ótimo

Fixe uma tarefa j, e um possível valor de makespan T.

Se escalonamento a possui makespan $\leq T$, então a deve designar cada tarefa em $\{1, \ldots, j\}$ a uma máquina i tal que $T \geq w_j/s_i$.

Defina $i(j, T) = \max\{i : T \geq w_j/s_i\}$, isto é, a máquina mais lenta que não viola a restrição para j (a tarefa mais leve).

Então, qualquer escalonamento cujo makespan não é superior a T associa as tarefas em $\{1, \ldots, j\}$ a máquinas em $\{1, \ldots, i(j, T)\}$.

Estimando makespan ótimo

Qualquer escalonamento cujo makespan não é superior a T associa as tarefas em $\{1, \ldots, j\}$ a máquinas em $\{1, \ldots, i(j, T)\}$.

Estimando makespan ótimo

Qualquer escalonamento cujo makespan não é superior a T associa as tarefas em $\{1, \ldots, j\}$ a máquinas em $\{1, \ldots, i(j, T)\}$.

Segue que, para cada j,

$$T \geq \frac{\sum_{k=1}^{j} w_k}{\sum_{\ell=1}^{i(j,T)} s_\ell}.$$

Estimando makespan ótimo

Qualquer escalonamento cujo makespan não é superior a T associa as tarefas em $\{1, \ldots, j\}$ a máquinas em $\{1, \ldots, i(j, T)\}$.

Segue que, para cada j,

$$T \geq \frac{\sum_{k=1}^{j} w_k}{\sum_{\ell=1}^{i(j,T)} s_\ell}.$$

Defina, agora,

$$T_j = \min_i \max \left\{ \frac{w_j}{s_i}, \frac{\sum_{k=1}^{j} w_k}{\sum_{\ell=1}^{i} s_\ell} \right\}.$$

Estimando makespan ótimo

Lema

Para qualquer tarefa j, o makespan ótimo não é inferior a T_j.

Estimando makespan ótimo

Lema

Para qualquer tarefa j, o makespan ótimo não é inferior a T_j.

Conseguimos uma boa cota inferior para o makespan ótimo:

$$T_{\mathsf{LB}} = \max_j T_j.$$

Um algoritmo fracionário

Algoritmo fracionário

Começamos com um escalonamento fracionário.

Um algoritmo fracionário

Algoritmo fracionário

Começamos com um escalonamento fracionário.

Se máquina i recebe uma proporção α da tarefa j, definimos a carga resultante como $(\alpha \cdot w_j)/s_i$.

Um algoritmo fracionário

Algoritmo fracionário

Começamos com um escalonamento fracionário.

Se máquina i recebe uma proporção α da tarefa j, definimos a carga resultante como $(\alpha \cdot w_j)/s_i$.

Em algum momento teremos que arredondar esse escalonamento.