Algoritmos e Jogos Cooperativos

WoPOCA 2018

Rafael C. S. Schouery
rafael@ic.unicamp.br

Universidade Estadual de Campinas

Introdução

O que é a Teoria dos Jogos?

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação
- Entender como alguém agirá em uma certa situação

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação
- Entender como alguém agirá em uma certa situação
- Projetar regras que incentivem um certo comportamento

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação
- Entender como alguém agirá em uma certa situação
- Projetar regras que incentivem um certo comportamento

O que é Teoria dos Jogos Algorítmica?

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação
- Entender como alguém agirá em uma certa situação
- Projetar regras que incentivem um certo comportamento

O que é Teoria dos Jogos Algorítmica?

- Interface entre a Teoria dos Jogos e a Computação

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação
- Entender como alguém agirá em uma certa situação
- Projetar regras que incentivem um certo comportamento

O que é Teoria dos Jogos Algorítmica?

- Interface entre a Teoria dos Jogos e a Computação
- Problemas da Economia do ponto de vista algorítmico

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação
- Entender como alguém agirá em uma certa situação
- Projetar regras que incentivem um certo comportamento

O que é Teoria dos Jogos Algorítmica?

- Interface entre a Teoria dos Jogos e a Computação
- Problemas da Economia do ponto de vista algorítmico
- Problemas da Computação do ponto de vista econômico

Introdução

O que é a Teoria dos Jogos?

- Estudo da interação entre agentes e dos resultados que possam ocorrer a partir dessa interação
- Entender como alguém agirá em uma certa situação
- Projetar regras que incentivem um certo comportamento

O que é Teoria dos Jogos Algorítmica?

- Interface entre a Teoria dos Jogos e a Computação
- Problemas da Economia do ponto de vista algorítmico
- Problemas da Computação do ponto de vista econômico
- Novos problemas, principalmente advindos da Internet

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos:

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos:

- Jogadores precisam cooperar para fazer algo

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos:

- Jogadores precisam cooperar para fazer algo
  - As ações individuais não são consideradas

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos:

- Jogadores precisam cooperar para fazer algo
  - As ações individuais não são consideradas
  - O que importa é o resultado atingido pela coletividade

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos:

- Jogadores precisam cooperar para fazer algo
  - As ações individuais não são consideradas
  - O que importa é o resultado atingido pela coletividade
- Mas os jogadores precisam ser incentivados a participar

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos:

- Jogadores precisam cooperar para fazer algo
  - As ações individuais não são consideradas
  - O que importa é o resultado atingido pela coletividade
- Mas os jogadores precisam ser incentivados a participar
  - O resultado precisa ser “justo”

Jogos Cooperativos vs. Não-Cooperativos

Jogos Não-Cooperativos:

- Resultado depende das escolhas feitas pelos agentes
  - Cada agente escolhe uma ação/estratégia
  - O resultado é a combinação das escolhas
- O agente é egoísta
  - Faz o que é melhor para si
  - Sem se preocupar com o resultado dos outros
- Ex: Pedra-Papel-Tesoura, Dilema do Prisioneiro, etc

Jogos Cooperativos:

- Jogadores precisam cooperar para fazer algo
  - As ações individuais não são consideradas
  - O que importa é o resultado atingido pela coletividade
- Mas os jogadores precisam ser incentivados a participar
  - O resultado precisa ser “justo”
- Ex: Doações de rins, dividir custos, etc

Sobre o Curso

Aula 1 - Cooperação sem dinheiro:

Sobre o Curso

Aula 1 - Cooperação sem dinheiro:

- Como fazer um mercado de doação de rins?

Sobre o Curso

Aula 1 - Cooperação sem dinheiro:

- Como fazer um mercado de doação de rins?
- Como escolher qual aluno é aceito em qual universidade?

Sobre o Curso

Aula 1 - Cooperação sem dinheiro:

- Como fazer um mercado de doação de rins?
- Como escolher qual aluno é aceito em qual universidade?
- Baseado em slides da Cris (IME/USP) com modificações

Sobre o Curso

Aula 1 - Cooperação sem dinheiro:

- Como fazer um mercado de doação de rins?
- Como escolher qual aluno é aceito em qual universidade?
- Baseado em slides da Cris (IME/USP) com modificações

Aula 2 - Cooperação com dinheiro:

Sobre o Curso

Aula 1 - Cooperação sem dinheiro:

- Como fazer um mercado de doação de rins?
- Como escolher qual aluno é aceito em qual universidade?
- Baseado em slides da Cris (IME/USP) com modificações

Aula 2 - Cooperação com dinheiro:

- Como dividir custo de maneira “justa”?

Sobre o Curso

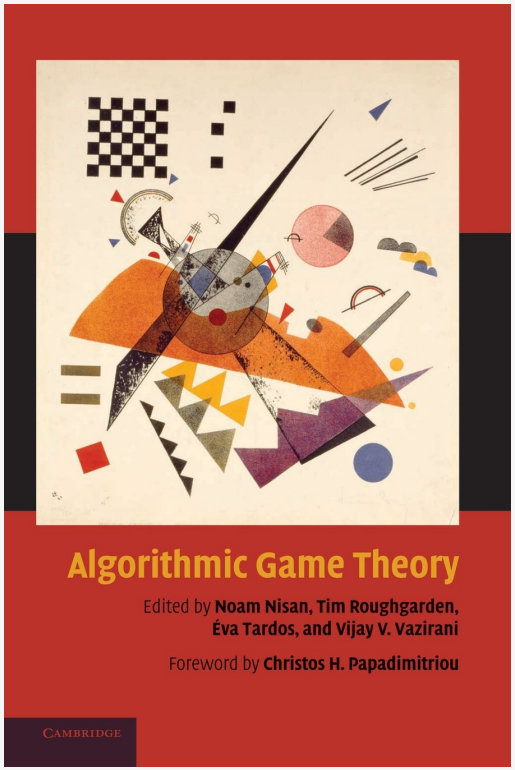
Aula 1 - Cooperação sem dinheiro:

- Como fazer um mercado de doação de rins?
- Como escolher qual aluno é aceito em qual universidade?
- Baseado em slides da Cris (IME/USP) com modificações

Aula 2 - Cooperação com dinheiro:

- Como dividir custo de maneira “justa”?
- Ex: criação de uma rede de telecomunicações

Bibliografia



N. Nisan, T. Roughgarden, E. Tardos e V. V. Vazirani, editores. Algorithmic Game Theory, Cambridge University Press, 2007.

`http://bit.ly/livro-agt`

Bibliografia

R. C. S. Schouery, O. Lee, F. K. Miyazawa, e E. C. Xavier. Tópicos da teoria dos jogos em computação, Editora do IMPA, 2015.

`http://bit.ly/slmx-ttjc`

Apenas conteúdo da aula 2

Alocação de Casas

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

- cada um começa com uma única casa

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

- cada um começa com uma única casa
- e tem uma ordem de preferência sobre todas as casas

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

- cada um começa com uma única casa
- e tem uma ordem de preferência sobre todas as casas

Objetivo: realocar as casas

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

- cada um começa com uma única casa
- e tem uma ordem de preferência sobre todas as casas

Objetivo: realocar as casas

- de modo que seja "justo"

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

- cada um começa com uma única casa
- e tem uma ordem de preferência sobre todas as casas

Objetivo: realocar as casas

- de modo que seja "justo"

Uma casa aqui representa um bem indivisível qualquer

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

- cada um começa com uma única casa
- e tem uma ordem de preferência sobre todas as casas

Objetivo: realocar as casas

- de modo que seja "justo"

Uma casa aqui representa um bem indivisível qualquer

- Al Roth criou um sistema de trocas de rins

Alocação de casas

Um conjunto N de agentes que desejam trocar casas entre si

- cada um começa com uma única casa
- e tem uma ordem de preferência sobre todas as casas

Objetivo: realocar as casas

- de modo que seja "justo"

Uma casa aqui representa um bem indivisível qualquer

- Al Roth criou um sistema de trocas de rins
- Um dos motivos de ganhar o Nobel em 2012

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

- i poderá denotar uma casa ou um agente

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

- i poderá denotar uma casa ou um agente
  - Ficará claro pelo contexto

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

- i poderá denotar uma casa ou um agente
  - Ficará claro pelo contexto

Uma alocação a é uma permutação de N

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

- i poderá denotar uma casa ou um agente
  - Ficará claro pelo contexto

Uma alocação a é uma permutação de N

- a_i é a casa que fica com o agente i

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

- i poderá denotar uma casa ou um agente
  - Ficará claro pelo contexto

Uma alocação a é uma permutação de N

- a_i é a casa que fica com o agente i
- A é conjunto de todas as alocações

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

- i poderá denotar uma casa ou um agente
  - Ficará claro pelo contexto

Uma alocação a é uma permutação de N

- a_i é a casa que fica com o agente i
- A é conjunto de todas as alocações

$\succ_i$ é a ordem de preferência do agente i

Alocação de casas (Scarf e Shapley'74)

O agente i começa com a casa i

- i poderá denotar uma casa ou um agente
  - Ficará claro pelo contexto

Uma alocação a é uma permutação de N

- a_i é a casa que fica com o agente i
- A é conjunto de todas as alocações

$\succ_i$ é a ordem de preferência do agente i

- $x \succ_i y$ significa que i prefere a casa x do que a casa y

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde
- pelo menos um jogador de S melhora

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde
- pelo menos um jogador de S melhora
  - existe $j \in S$ com $z_j \succ_j a_j$

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde
- pelo menos um jogador de S melhora
  - existe $j \in S$ com $z_j \succ_j a_j$
- e ninguém de S piora

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde
- pelo menos um jogador de S melhora
  - existe $j \in S$ com $z_j \succ_j a_j$
- e ninguém de S piora
  - para todo $i \in S$, $z_i \succeq_i a_i$

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde
- pelo menos um jogador de S melhora
  - existe $j \in S$ com $z_j \succ_j a_j$
- e ninguém de S piora
  - para todo $i \in S$, $z_i \succeq_i a_i$

Núcleo: conjunto de alocações sem coalizão bloqueadora

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde
- pelo menos um jogador de S melhora
  - existe $j \in S$ com $z_j \succ_j a_j$
- e ninguém de S piora
  - para todo $i \in S$, $z_i \succeq_i a_i$

Núcleo: conjunto de alocações sem coalizão bloqueadora

- Ou seja, queremos achar uma alocação no núcleo

Coalizão

$S \subseteq N$ é uma coalizão bloqueadora para uma alocação a se:

- os jogadores de S podem trocar apenas entre eles
- de forma que alguém de S melhora em relação a a
- e ninguém piora de S em relação a a

Para $S \subseteq N$, seja $A(S) = \{a \in A : a_i \in S, \forall i \in S\}$

- $A(S)$: alocações onde agentes de S trocam casas entre si

S é uma coalizão bloqueadora para uma alocação a

- se existe $z \in A(S)$ onde
- pelo menos um jogador de S melhora
  - existe $j \in S$ com $z_j \succ_j a_j$
- e ninguém de S piora
  - para todo $i \in S$, $z_i \succeq_i a_i$

Núcleo: conjunto de alocações sem coalizão bloqueadora

- Ou seja, queremos achar uma alocação no núcleo
- se é que existe uma tal alocação...

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

1. Inicialize N como o conjunto de todos agentes

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

1. Inicialize N como o conjunto de todos agentes
2. Enquanto N não for vazio

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

1. Inicialize N como o conjunto de todos agentes
2. Enquanto N não for vazio
   2.1 Construa o grafo $G(N)$

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

1. Inicialize N como o conjunto de todos agentes
2. Enquanto N não for vazio
   - 2.1 Construa o grafo $G(N)$
   - 2.2 Seja A o conjunto de agente em circuitos de G

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

1. Inicialize N como o conjunto de todos agentes
2. Enquanto N não for vazio
   - 2.1 Construa o grafo $G(N)$
   - 2.2 Seja A o conjunto de agente em circuitos de G
   - 2.3 Troque as casas entre os membros de A de acordo com G

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

1. Inicialize N como o conjunto de todos agentes
2. Enquanto N não for vazio
   - 2.1 Construa o grafo $G(N)$
   - 2.2 Seja A o conjunto de agente em circuitos de G
   - 2.3 Troque as casas entre os membros de A de acordo com G
   - 2.4 Faça $N = N \setminus A$

Algoritmo TTC - top trading cycle

Para um conjunto N' de agentes, o digrafo $G(N')$:

- tem um vértice para cada agente
- tem arco de i para j se a casa de j é a favorita de i

Cada vértice de $G(N')$ tem grau de saída 1

- há exatamente um circuito em cada componente conexo
- isto é, os circuitos são vértice-disjuntos

Top Trading Cycle (David Gale)

1. Inicialize N como o conjunto de todos agentes
2. Enquanto N não for vazio
   - 2.1 Construa o grafo $G(N)$
   - 2.2 Seja A o conjunto de agente em circuitos de G
   - 2.3 Troque as casas entre os membros de A de acordo com G
   - 2.4 Faça $N = N \setminus A$

Vamos denotar por N_i os agentes que sairam na iteração i

Núcleo

Teorema: O núcleo do problema da alocação de casas consiste exatamente de uma alocação

Núcleo

Teorema: O núcleo do problema da alocação de casas consiste exatamente de uma alocação

Prova: Primeiro, vamos mostrar que se uma alocação está no núcleo ela precisa ser a devolvida pelo algoritmo TTC

Núcleo

Teorema: O núcleo do problema da alocação de casas consiste exatamente de uma alocação

Prova: Primeiro, vamos mostrar que se uma alocação está no núcleo ela precisa ser a devolvida pelo algoritmo TTC

Em uma alocação no núcleo, os agentes de N_j precisam receber a mesma casa dada pelo algoritmo:

Núcleo

Teorema: O núcleo do problema da alocação de casas consiste exatamente de uma alocação

Prova: Primeiro, vamos mostrar que se uma alocação está no núcleo ela precisa ser a devolvida pelo algoritmo TTC

Em uma alocação no núcleo, os agentes de N_j precisam receber a mesma casa dada pelo algoritmo:

- Prova por indução forte em j

Núcleo

Teorema: O núcleo do problema da alocação de casas consiste exatamente de uma alocação

Prova: Primeiro, vamos mostrar que se uma alocação está no núcleo ela precisa ser a devolvida pelo algoritmo TTC

Em uma alocação no núcleo, os agentes de N_j precisam receber a mesma casa dada pelo algoritmo:

- Prova por indução forte em j
- Suponha que todos os agentes de $\bigcup_{k<j} N_k$ recebam a mesma casa que a dada pelo algoritmo

Núcleo

Teorema: O núcleo do problema da alocação de casas consiste exatamente de uma alocação

Prova: Primeiro, vamos mostrar que se uma alocação está no núcleo ela precisa ser a devolvida pelo algoritmo TTC

Em uma alocação no núcleo, os agentes de N_j precisam receber a mesma casa dada pelo algoritmo:

- Prova por indução forte em j
- Suponha que todos os agentes de $\bigcup_{k<j} N_k$ recebam a mesma casa que a dada pelo algoritmo
- Assim, agentes de N_j só podem receber casas piores ou iguais as dadas pelo algoritmo

Núcleo

Teorema: O núcleo do problema da alocação de casas consiste exatamente de uma alocação

Prova: Primeiro, vamos mostrar que se uma alocação está no núcleo ela precisa ser a devolvida pelo algoritmo TTC

Em uma alocação no núcleo, os agentes de N_j precisam receber a mesma casa dada pelo algoritmo:

- Prova por indução forte em j
- Suponha que todos os agentes de $\bigcup_{k<j} N_k$ recebam a mesma casa que a dada pelo algoritmo
- Assim, agentes de N_j só podem receber casas piores ou iguais as dadas pelo algoritmo
- Qualquer alocação em que $i \in N_j$ recebe uma casa pior do que a do algoritmo tem N_j como coalisão bloqueadora

Núcleo

Ou seja, se existe uma alocação no núcleo, ela precisa ser igual a alocação encontrada pelo algoritmo TTC

Núcleo

Ou seja, se existe uma alocação no núcleo, ela precisa ser igual a alocação encontrada pelo algoritmo TTC

Mas será que a alocação a do algoritmo está no núcleo?

Núcleo

Ou seja, se existe uma alocação no núcleo, ela precisa ser igual a alocação encontrada pelo algoritmo TTC

Mas será que a alocação a do algoritmo está no núcleo?

Suponha que não e seja S é uma coalizão bloqueadora para a

Núcleo

Ou seja, se existe uma alocação no núcleo, ela precisa ser igual a alocação encontrada pelo algoritmo TTC

Mas será que a alocação a do algoritmo está no núcleo?

Suponha que não e seja S é uma coalizão bloqueadora para a

Então, existe z em $A(S)$ tal que:

Núcleo

Ou seja, se existe uma alocação no núcleo, ela precisa ser igual a alocação encontrada pelo algoritmo TTC

Mas será que a alocação a do algoritmo está no núcleo?

Suponha que não e seja S é uma coalizão bloqueadora para a

Então, existe z em $A(S)$ tal que:

- existe $j \in S$ com $z_j \succ_j a_j$

Núcleo

Ou seja, se existe uma alocação no núcleo, ela precisa ser igual a alocação encontrada pelo algoritmo TTC

Mas será que a alocação a do algoritmo está no núcleo?

Suponha que não e seja S é uma coalizão bloqueadora para a

Então, existe z em $A(S)$ tal que:

- existe $j \in S$ com $z_j \succ_j a_j$
- para todo $i \neq j$ em S, ou $z_i \succ_i a_i$ ou $z_i = a_i$

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

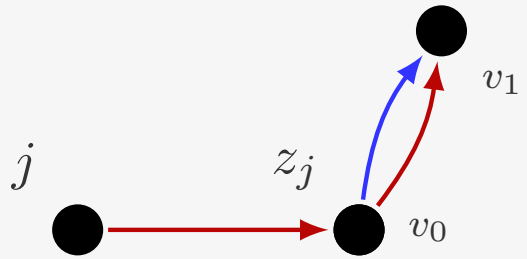
- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

Temos que $v_0 := z_j$ está em N_r com $r < k$

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

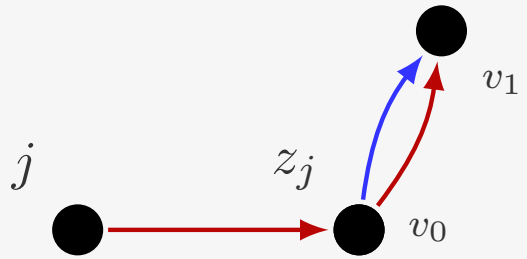


Assim, v_0 recebe a mesma casa em a e z (i.e. $a_{v_0} = z_{v_0}$)

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

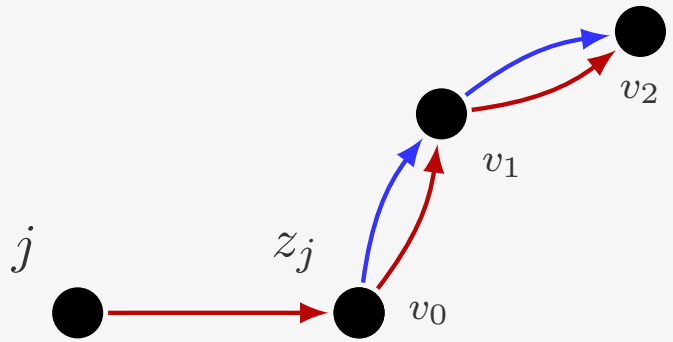


E, portanto, $v_1 := a_{v_0}$ também está em $S \cap N_r$

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

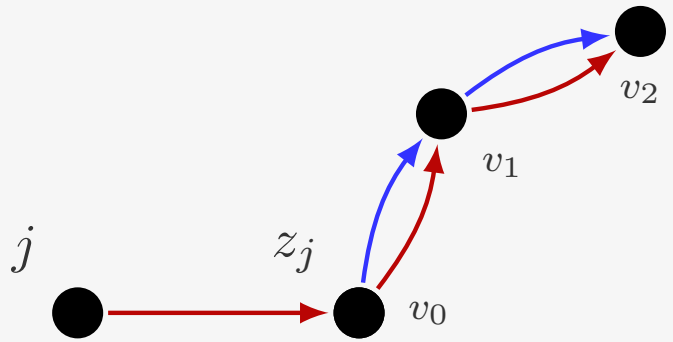


Assim, v_1 recebe a mesma casa em a e z (i.e. $a_{v_1} = z_{v_1}$)

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

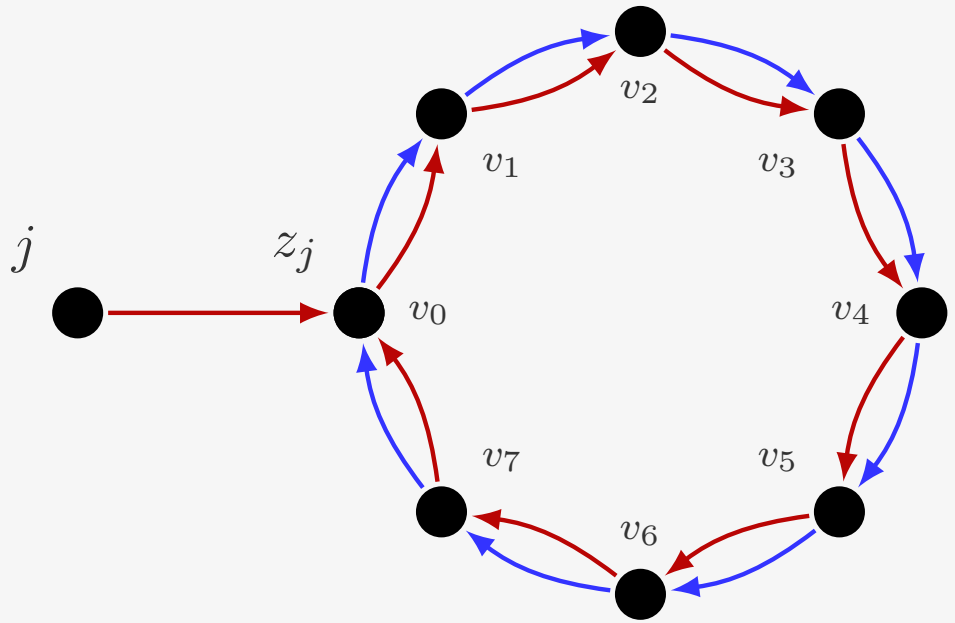


E, portanto, $v_2 := a_{v_1}$ também está em $S \cap N_r$

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

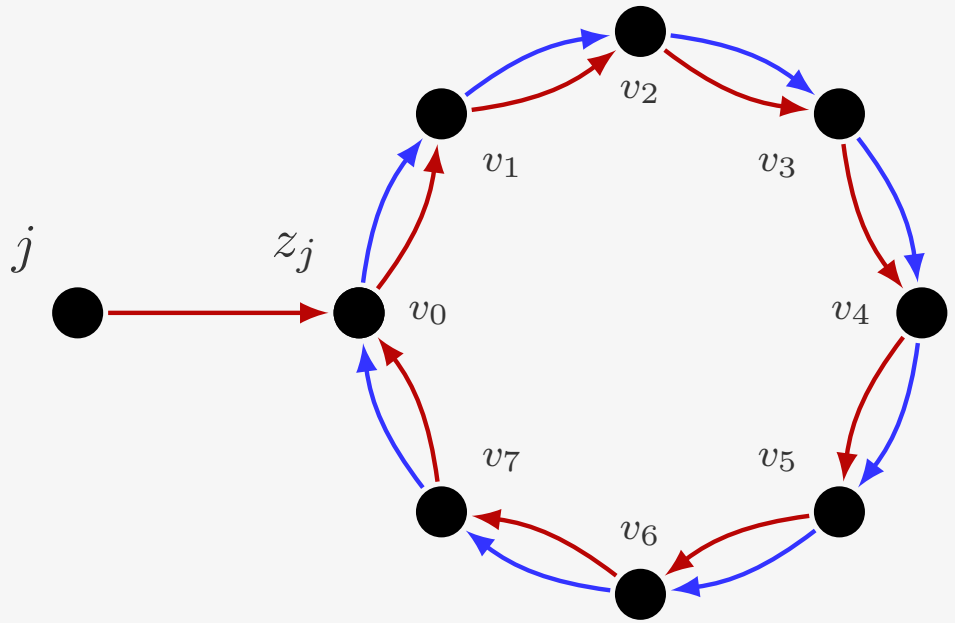


Seguindo o raciocínio, temos que v_n está em $S \cap N_r$

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j

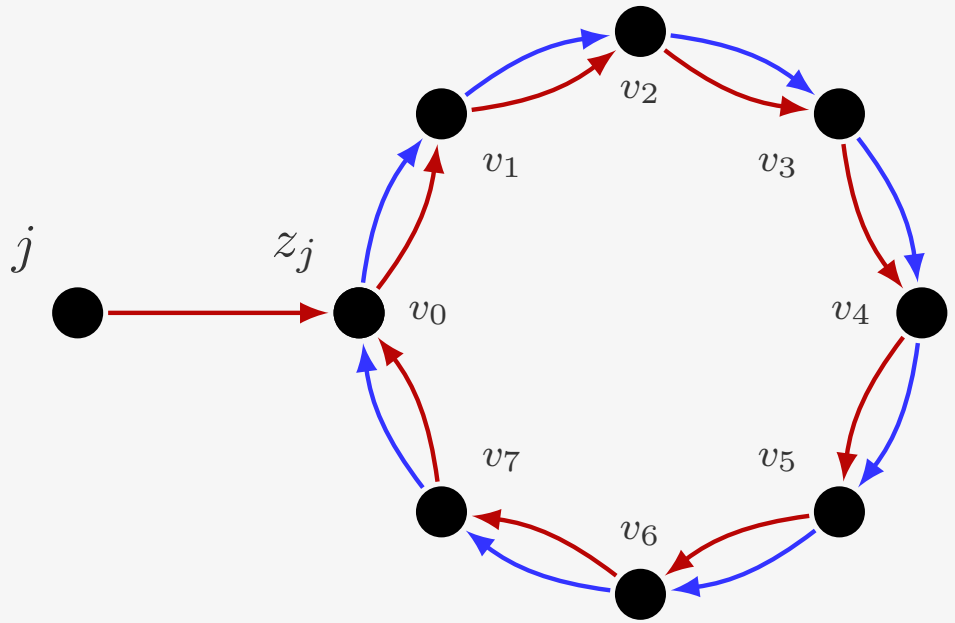


Assim, v_n recebe a mesma casa em a e z (i.e. $a_{v_n} = z_{v_n}$)

Núcleo

Escolha $j \in N_k$ com $z_j \succ_j a_j$ (i.e. que melhorou) e k mínimo

- Como j não ficou com z_j, ela foi alocada antes do turno k
- Senão, j apontaria para z_j no turno k e não para a_j



Mas, $z_{v_n} = v_j$, contradição com z ser uma alocação

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

- O algoritmo se baseia nas preferências dos agentes

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

- O algoritmo se baseia nas preferências dos agentes
- Essas preferências são reportadas pelos agentes

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

- O algoritmo se baseia nas preferências dos agentes
- Essas preferências são reportadas pelos agentes
- Um agente poderia mentir para tentar ser beneficiado

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

- O algoritmo se baseia nas preferências dos agentes
- Essas preferências são reportadas pelos agentes
- Um agente poderia mentir para tentar ser beneficiado

De forma geral, dizemos que é algo é à prova de estratégia se mentir não pode beneficiar o agente

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

- O algoritmo se baseia nas preferências dos agentes
- Essas preferências são reportadas pelos agentes
- Um agente poderia mentir para tentar ser beneficiado

De forma geral, dizemos que é algo é à prova de estratégia se mentir não pode beneficiar o agente

- Isto é, reportar uma ordem de preferência diferente

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

- O algoritmo se baseia nas preferências dos agentes
- Essas preferências são reportadas pelos agentes
- Um agente poderia mentir para tentar ser beneficiado

De forma geral, dizemos que é algo é à prova de estratégia se mentir não pode beneficiar o agente

- Isto é, reportar uma ordem de preferência diferente
- não faz com que ele ganhe uma casa melhor no final

Mentindo para melhorar

Será que é possível mentir para melhorar a casa recebida?

- O algoritmo se baseia nas preferências dos agentes
- Essas preferências são reportadas pelos agentes
- Um agente poderia mentir para tentar ser beneficiado

De forma geral, dizemos que é algo é à prova de estratégia se mentir não pode beneficiar o agente

- Isto é, reportar uma ordem de preferência diferente
- não faz com que ele ganhe uma casa melhor no final
- de acordo com a real ordem de preferência do agente

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova:

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π : as reais preferências do agentes

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π: as reais preferências do agentes
- a: alocação devolvida pelo TTC com entrada π

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π: as reais preferências do agentes
- a: alocação devolvida pelo TTC com entrada π
- π': preferências quando um agente j mente

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π : as reais preferências do agentes
- a : alocação devolvida pelo TTC com entrada π
- π' : preferências quando um agente j mente
- a' : alocação devolvida pelo TTC com entrada π'

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π : as reais preferências do agentes
- a : alocação devolvida pelo TTC com entrada π
- π' : preferências quando um agente j mente
- a' : alocação devolvida pelo TTC com entrada π'

Se o TTC não é à prova de estratégia, então $a'_j \succ_j a_j$

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π : as reais preferências do agentes
- a : alocação devolvida pelo TTC com entrada π
- π' : preferências quando um agente j mente
- a' : alocação devolvida pelo TTC com entrada π'

Se o TTC não é à prova de estratégia, então $a'_j \succ_j a_j$

Suponha que, quando fala a verdade, $j \in N_k$

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π: as reais preferências do agentes
- a: alocação devolvida pelo TTC com entrada π
- π': preferências quando um agente j mente
- a': alocação devolvida pelo TTC com entrada π'

Se o TTC não é à prova de estratégia, então $a'_j \succ_j a_j$

Suponha que, quando fala a verdade, $j \in N_k$

- Todos os circuitos que formados até o passo k com π continuam sendo formados com π'

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π: as reais preferências do agentes
- a: alocação devolvida pelo TTC com entrada π
- π': preferências quando um agente j mente
- a': alocação devolvida pelo TTC com entrada π'

Se o TTC não é à prova de estratégia, então $a'_j \succ_j a_j$

Suponha que, quando fala a verdade, $j \in N_k$

- Todos os circuitos que formados até o passo k com π continuam sendo formados com π'
- i.e. $a_i = a'_i$ para todo $i \in \bigcup_{r=1}^{k-1} N_r$

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π: as reais preferências do agentes
- a: alocação devolvida pelo TTC com entrada π
- π': preferências quando um agente j mente
- a': alocação devolvida pelo TTC com entrada π'

Se o TTC não é à prova de estratégia, então $a'_j \succ_j a_j$

Suponha que, quando fala a verdade, $j \in N_k$

- Todos os circuitos que formados até o passo k com π continuam sendo formados com π'
- i.e. $a_i = a'_i$ para todo $i \in \bigcup_{r=1}^{k-1} N_r$
- portanto $a'_j \in N \setminus \bigcup_{r=1}^{k-1} N_r$

TTC é à prova de estratégia

Teorema: O algoritmo TTC é à prova de estratégia

Prova: Seja

- π: as reais preferências do agentes
- a: alocação devolvida pelo TTC com entrada π
- π': preferências quando um agente j mente
- a': alocação devolvida pelo TTC com entrada π'

Se o TTC não é à prova de estratégia, então $a'_j \succ_j a_j$

Suponha que, quando fala a verdade, $j \in N_k$

- Todos os circuitos que formados até o passo k com π continuam sendo formados com π'
- i.e. $a_i = a'_i$ para todo $i \in \bigcup_{r=1}^{k-1} N_r$
- portanto $a'_j \in N \setminus \bigcup_{r=1}^{k-1} N_r$

Mas, o algoritmo já dá para j a melhor casa de $N \setminus \bigcup_{r=1}^{k-1} N_r$

Emparelhamentos estáveis

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

- escolher quais estudantes preencherão quais vagas

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

- escolher quais estudantes preencherão quais vagas
- atribuir residentes aos hospitais

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

- escolher quais estudantes preencherão quais vagas
- atribuir residentes aos hospitais

Devemos levar em conta as preferência individuais

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

- escolher quais estudantes preencherão quais vagas
- atribuir residentes aos hospitais

Devemos levar em conta as preferência individuais

- estudantes tem uma lista de preferência das universidades

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

- escolher quais estudantes preencherão quais vagas
- atribuir residentes aos hospitais

Devemos levar em conta as preferência individuais

- estudantes tem uma lista de preferência das universidades
- universidades tem uma lista de preferência de estudantes

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

- escolher quais estudantes preencherão quais vagas
- atribuir residentes aos hospitais

Devemos levar em conta as preferência individuais

- estudantes tem uma lista de preferência das universidades
- universidades tem uma lista de preferência de estudantes

E queremos que o emparelhamento seja "bom" (estável)...

Emparelhamentos estáveis (Gale e Shapley'62)

Temos dois grupos distintos:

- estudantes e universidades
- médicos residentes e hospitais

Queremos emparelhar os dois grupos

- escolher quais estudantes preencherão quais vagas
- atribuir residentes aos hospitais

Devemos levar em conta as preferência individuais

- estudantes tem uma lista de preferência das universidades
- universidades tem uma lista de preferência de estudantes

E queremos que o emparelhamento seja "bom" (estável)...

- não há um par estudante-universidade que quer desviar

Emparelhamentos estáveis

Temos:

Emparelhamentos estáveis

Temos:

- H : conjunto de homens

Emparelhamentos estáveis

Temos:

- H : conjunto de homens
- M : conjunto de mulheres

Emparelhamentos estáveis

Temos:

- H : conjunto de homens
- M : conjunto de mulheres
- Cada homem tem uma ordem de preferência sobre M

Emparelhamentos estáveis

Temos:

- H : conjunto de homens
- M : conjunto de mulheres
- Cada homem tem uma ordem de preferência sobre M
- Cada mulher tem uma ordem de preferência sobre H

Emparelhamentos estáveis

Temos:

- H : conjunto de homens
- M : conjunto de mulheres
- Cada homem tem uma ordem de preferência sobre M
- Cada mulher tem uma ordem de preferência sobre H
- Adicione homem/mulher fictício para representar a possibilidade de ficar solteiro

Emparelhamentos estáveis

Temos:

- H : conjunto de homens
- M : conjunto de mulheres
- Cada homem tem uma ordem de preferência sobre M
- Cada mulher tem uma ordem de preferência sobre H
- Adicione homem/mulher fictício para representar a possibilidade de ficar solteiro
  - Não vou fazer isso nos exemplos para não ficar grande...

Emparelhamentos estáveis

Temos:

- H: conjunto de homens
- M: conjunto de mulheres
- Cada homem tem uma ordem de preferência sobre M
- Cada mulher tem uma ordem de preferência sobre H
- Adicione homem/mulher fictício para representar a possibilidade de ficar solteiro
  - Não vou fazer isso nos exemplos para não ficar grande...

Assim $|H| = |M|$

Emparelhamentos estáveis

Temos:

- H: conjunto de homens
- M: conjunto de mulheres
- Cada homem tem uma ordem de preferência sobre M
- Cada mulher tem uma ordem de preferência sobre H
- Adicione homem/mulher fictício para representar a possibilidade de ficar solteiro
  - Não vou fazer isso nos exemplos para não ficar grande...

Assim $|H| = |M|$

Emparelhamento de H em M

Emparelhamentos estáveis

Temos:

- H : conjunto de homens
- M : conjunto de mulheres
- Cada homem tem uma ordem de preferência sobre M
- Cada mulher tem uma ordem de preferência sobre H
- Adicione homem/mulher fictício para representar a possibilidade de ficar solteiro
  - Não vou fazer isso nos exemplos para não ficar grande...

Assim $|H| = |M|$

Emparelhamento de H em M

- cada homem casado exatamente com uma mulher

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$
- m está emparelhada com $h' \neq m$

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$
- m está emparelhada com $h' \neq m$
- mas $m \succ_h m'$ e $h \succ_m h'$

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$
- m está emparelhada com $h' \neq m$
- mas $m \succ_h m'$ e $h \succ_m h'$

Neste caso, h e m preferiam se casar um com o outro

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$
- m está emparelhada com $h' \neq m$
- mas $m \succ_h m'$ e $h \succ_m h'$

Neste caso, h e m preferiam se casar um com o outro

- Podem ignorar o que foi proposto

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$
- m está emparelhada com $h' \neq m$
- mas $m \succ_h m'$ e $h \succ_m h'$

Neste caso, h e m preferiam se casar um com o outro

- Podem ignorar o que foi proposto
- e fazer um acordo entre eles

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$
- m está emparelhada com $h' \neq m$
- mas $m \succ_h m'$ e $h \succ_m h'$

Neste caso, h e m preferiam se casar um com o outro

- Podem ignorar o que foi proposto
- e fazer um acordo entre eles
- O par (h, m) é um par bloqueador

Emparelhamentos estáveis

Um emparelhamento é instável se existem um homem h e uma mulher m tal que

- h está emparelhado com $m' \neq m$
- m está emparelhada com $h' \neq m$
- mas $m \succ_h m'$ e $h \succ_m h'$

Neste caso, h e m preferiam se casar um com o outro

- Podem ignorar o que foi proposto
- e fazer um acordo entre eles
- O par (h, m) é um par bloqueador

Um emparelhamento é estável se não tem par bloqueador

Emparelhamentos estáveis: exemplo

Ordens de preferências para $n = 3$:

Emparelhamentos estáveis: exemplo

Ordens de preferências para $n = 3$:

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Emparelhamentos estáveis: exemplo

Ordens de preferências para $n = 3$:

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

O emparelhamento $\{(h_1, m_1), (h_2, m_2), (h_3, m_3)\}$ é instável, pois (h_1, m_2) é um par bloqueador

Emparelhamentos estáveis: exemplo

Ordens de preferências para $n = 3$:

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

O emparelhamento $\{(h_1, m_1), (h_2, m_2), (h_3, m_3)\}$ é instável, pois (h_1, m_2) é um par bloqueador

Já o emparelhamento $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$ é estável

Emparelhamentos estáveis: exemplo

Ordens de preferências para $n = 3$:

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

O emparelhamento $\{(h_1, m_1), (h_2, m_2), (h_3, m_3)\}$ é instável, pois (h_1, m_2) é um par bloqueador

Já o emparelhamento $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$ é estável

Sempre existe emparelhamento estável?

Emparelhamentos estáveis: exemplo

Ordens de preferências para $n = 3$:

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

O emparelhamento $\{(h_1, m_1), (h_2, m_2), (h_3, m_3)\}$ é instável, pois (h_1, m_2) é um par bloqueador

Já o emparelhamento $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$ é estável

Sempre existe emparelhamento estável?

- Se sim, como encontrá-lo?

Algoritmo da aceitação postergada

Versão com proposta masculina

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista
- Cada mulher que recebeu mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista
- Cada mulher que recebeu mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Para esse, posterga a sua resposta

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista
- Cada mulher que recebeu mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Para esse, posterga a sua resposta

Nova rodada:

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista
- Cada mulher que recebeu mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Para esse, posterga a sua resposta

Nova rodada:

- Cada homem que teve sua proposta recusada propõe à próxima mulher de sua lista

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista
- Cada mulher que recebeu mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Para esse, posterga a sua resposta

Nova rodada:

- Cada homem que teve sua proposta recusada propõe à próxima mulher de sua lista
- Cada mulher com mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista
- Cada mulher que recebeu mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Para esse, posterga a sua resposta

Nova rodada:

- Cada homem que teve sua proposta recusada propõe à próxima mulher de sua lista
- Cada mulher com mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Pode recusar proposta recebida em rodada anterior

Algoritmo da aceitação postergada

Versão com proposta masculina

Primeira rodada:

- Cada homem propõe à primeira mulher de sua lista
- Cada mulher que recebeu mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Para esse, posterga a sua resposta

Nova rodada:

- Cada homem que teve sua proposta recusada propõe à próxima mulher de sua lista
- Cada mulher com mais de uma proposta recusa todas, exceto a do homem preferido entre os candidatos
  - Pode recusar proposta recebida em rodada anterior

O processo termina em não mais que n^2 rodadas

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 :

m_2 :

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 :

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 : h_2

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1: h_2 h_3

m_2: h_1

m_3:

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : h_2 h_3

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : ~~h_2~~ h_3

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1: ~~h_2~~ h_3

m_2: h_1

m_3: h_2

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1: ~~h_2~~ h_3

m_2: h_1

m_3: h_2

Emparelhamento produzido: $\{(h_1, m_2), (h_2, m_3), (h_3, m_1)\}$

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1: ~~h_2~~ h_3

m_2: h_1

m_3: h_2

Emparelhamento produzido: $\{(h_1, m_2), (h_2, m_3), (h_3, m_1)\}$

Note que tal emparelhamento é estável!

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova:

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

- onde h_1 está emparelhado com m_2

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

- onde h_1 está emparelhado com m_2
- e m_1 está emparelhado com h_2

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

- onde h_1 está emparelhado com m_2
- e m_1 está emparelhado com h_2

Então:

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

- onde h_1 está emparelhado com m_2
- e m_1 está emparelhado com h_2

Então:

- como $m_1 \succ_{h_1} m_2$, h_1 propôs para m_1 antes de propor para m_2

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

- onde h_1 está emparelhado com m_2
- e m_1 está emparelhado com h_2

Então:

- como $m_1 \succ_{h_1} m_2$, h_1 propôs para m_1 antes de propor para m_2
- como h_1 ficou com m_2, m_1 trocou h_1 por um homem melhor

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

- onde h_1 está emparelhado com m_2
- e m_1 está emparelhado com h_2

Então:

- como $m_1 \succ_{h_1} m_2$, h_1 propôs para m_1 antes de propor para m_2
- como h_1 ficou com m_2, m_1 trocou h_1 por um homem melhor
- então $h_2 \succ_{m_1} h_1$

Estabilidade do emparelhamento

Teorema. O emparelhamento produzido pelo algoritmo da aceitação postergada é estável

Prova: Suponha que existe um par bloqueador (h_1, m_1)

- onde h_1 está emparelhado com m_2
- e m_1 está emparelhado com h_2

Então:

- como $m_1 \succ_{h_1} m_2$, h_1 propôs para m_1 antes de propor para m_2
- como h_1 ficou com m_2, m_1 trocou h_1 por um homem melhor
- então $h_2 \succ_{m_1} h_1$

ou seja, (h_1, m_1) não é um par bloqueador

Núcleo

Num emparelhamento estável nenhum (h, m) prefere sair do mecanismo e se casar

Núcleo

Num emparelhamento estável nenhum (h, m) prefere sair do mecanismo e se casar

- E se considerarmos um grupo que sai do mecanismo?

Núcleo

Num emparelhamento estável nenhum (h, m) prefere sair do mecanismo e se casar

- E se considerarmos um grupo que sai do mecanismo?

Um emparelhamento ν domina um emparelhamento μ se existe $S \subseteq H \cup M$ tal que para todo h e m em S, temos

Núcleo

Num emparelhamento estável nenhum (h, m) prefere sair do mecanismo e se casar

- E se considerarmos um grupo que sai do mecanismo?

Um emparelhamento ν domina um emparelhamento μ se existe $S \subseteq H \cup M$ tal que para todo h e m em S, temos

- $\nu(h)$ e $\nu(m)$ pertencem a S

Núcleo

Num emparelhamento estável nenhum (h, m) prefere sair do mecanismo e se casar

- E se considerarmos um grupo que sai do mecanismo?

Um emparelhamento ν domina um emparelhamento μ se existe $S \subseteq H \cup M$ tal que para todo h e m em S, temos

- $\nu(h)$ e $\nu(m)$ pertencem a S
- $\nu(h) \succ_h \mu(h)$ e $\nu(m) \succ_m \mu(m)$

Núcleo

Num emparelhamento estável nenhum (h, m) prefere sair do mecanismo e se casar

- E se considerarmos um grupo que sai do mecanismo?

Um emparelhamento ν domina um emparelhamento μ se existe $S \subseteq H \cup M$ tal que para todo h e m em S, temos

- $\nu(h)$ e $\nu(m)$ pertencem a S
- $\nu(h) \succ_h \mu(h)$ e $\nu(m) \succ_m \mu(m)$

Um emparelhamento μ está no núcleo se e somente se não existe emparelhamento ν que o domina

Núcleo

Num emparelhamento estável nenhum (h, m) prefere sair do mecanismo e se casar

- E se considerarmos um grupo que sai do mecanismo?

Um emparelhamento ν domina um emparelhamento μ se existe $S \subseteq H \cup M$ tal que para todo h e m em S, temos

- $\nu(h)$ e $\nu(m)$ pertencem a S
- $\nu(h) \succ_h \mu(h)$ e $\nu(m) \succ_m \mu(m)$

Um emparelhamento μ está no núcleo se e somente se não existe emparelhamento ν que o domina

Teorema. O núcleo do jogo de emparelhamento é o conjunto de todos os emparelhamentos estáveis

Estabilidade do emparelhamento

No exemplo, aplicando a versão da proposta feminina, obtemos o emparelhamento $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$.

Estabilidade do emparelhamento

No exemplo, aplicando a versão da proposta feminina, obtemos o emparelhamento $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$.

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Estabilidade do emparelhamento

No exemplo, aplicando a versão da proposta feminina, obtemos o emparelhamento $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$.

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Diferente do obtido pela proposta masculina!

Estabilidade do emparelhamento

No exemplo, aplicando a versão da proposta feminina, obtemos o emparelhamento $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$.

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Diferente do obtido pela proposta masculina!

Há alguma diferença significativa entre estes emparelhamentos?

Emparelhamentos ótimos

Emparelhamento ν é ótimo-masculino se

Emparelhamentos ótimos

Emparelhamento ν é ótimo-masculino se

- não há emparelhamento estável μ tal que

Emparelhamentos ótimos

Emparelhamento ν é ótimo-masculino se

- não há emparelhamento estável μ tal que
- existe j em H com $\mu(j) \succ_j \nu(j)$ e

Emparelhamentos ótimos

Emparelhamento ν é ótimo-masculino se

- não há emparelhamento estável μ tal que
- existe j em H com $\mu(j) \succ_j \nu(j)$ e
- para todo h em H, $\mu(h) \succeq_h \nu(h)$

Emparelhamentos ótimos

Emparelhamento ν é ótimo-masculino se

- não há emparelhamento estável μ tal que
- existe j em H com $\mu(j) \succ_j \nu(j)$ e
- para todo h em H, $\mu(h) \succeq_h \nu(h)$

Definição de emparelhamento ótimo-feminino é análoga

Emparelhamentos ótimos

Teorema: O algoritmo da aceitação postergada com proposta masculina produz um emparelhamento ótimo-masculino

Emparelhamentos ótimos

Teorema: O algoritmo da aceitação postergada com proposta masculina produz um emparelhamento ótimo-masculino

Prova:

Emparelhamentos ótimos

Teorema: O algoritmo da aceitação postergada com proposta masculina produz um emparelhamento ótimo-masculino

Prova: Seja μ o emparelhamento encontrado pelo algoritmo e suponha que μ não é ótimo-masculino

Emparelhamentos ótimos

Teorema: O algoritmo da aceitação postergada com proposta masculina produz um emparelhamento ótimo-masculino

Prova: Seja μ o emparelhamento encontrado pelo algoritmo e suponha que μ não é ótimo-masculino

Existe emparelhamento estável ν tal que existe j em H com $\nu(j) \succ_j \mu(j)$ e para todo h em H, $\nu(h) \succeq_h \mu(h)$

Emparelhamentos ótimos

Teorema: O algoritmo da aceitação postergada com proposta masculina produz um emparelhamento ótimo-masculino

Prova: Seja μ o emparelhamento encontrado pelo algoritmo e suponha que μ não é ótimo-masculino

Existe emparelhamento estável ν tal que existe j em H com $\nu(j) \succ_j \mu(j)$ e para todo h em H, $\nu(h) \succeq_h \mu(h)$

- j propôs primeiro para $\nu(j)$ e foi rejeitado

Emparelhamentos ótimos

Teorema: O algoritmo da aceitação postergada com proposta masculina produz um emparelhamento ótimo-masculino

Prova: Seja μ o emparelhamento encontrado pelo algoritmo e suponha que μ não é ótimo-masculino

Existe emparelhamento estável ν tal que existe j em H com $\nu(j) \succ_j \mu(j)$ e para todo h em H, $\nu(h) \succeq_h \mu(h)$

- j propôs primeiro para $\nu(j)$ e foi rejeitado
- escolha j como o primeiro homem que foi rejeitado pela mulher $\nu(j)$

Emparelhamentos ótimos

- j: primeiro homem que foi rejeitado pela mulher $\nu(j)$

Emparelhamentos ótimos

- j: primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Emparelhamentos ótimos

- j : primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Então:

Emparelhamentos ótimos

- j: primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Então:

- $\nu(j)$ recebe uma proposta de um homem i que ela prefere

Emparelhamentos ótimos

- j: primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Então:

- $\nu(j)$ recebe uma proposta de um homem i que ela prefere
- i prefere a mulher $\nu(j)$ à mulher $\nu(i)$

Emparelhamentos ótimos

- j: primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Então:

- $\nu(j)$ recebe uma proposta de um homem i que ela prefere
- i prefere a mulher $\nu(j)$ à mulher $\nu(i)$
  - caso contrário, i já teria sido rejeitado por $\nu(i)$

Emparelhamentos ótimos

- j : primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Então:

- $\nu(j)$ recebe uma proposta de um homem i que ela prefere
- i prefere a mulher $\nu(j)$ à mulher $\nu(i)$
  - caso contrário, i já teria sido rejeitado por $\nu(i)$
  - e não teríamos escolhido j

Emparelhamentos ótimos

- j: primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Então:

- $\nu(j)$ recebe uma proposta de um homem i que ela prefere
- i prefere a mulher $\nu(j)$ à mulher $\nu(i)$
  - caso contrário, i já teria sido rejeitado por $\nu(i)$
  - e não teríamos escolhido j
- portanto, $i \succ_{\nu(j)} j$ e $\nu(j) \succ_i \nu(i)$

Emparelhamentos ótimos

- j: primeiro homem que foi rejeitado pela mulher $\nu(j)$
- $\nu(j) \succ_j \mu(j)$

Então:

- $\nu(j)$ recebe uma proposta de um homem i que ela prefere
- i prefere a mulher $\nu(j)$ à mulher $\nu(i)$
  - caso contrário, i já teria sido rejeitado por $\nu(i)$
  - e não teríamos escolhido j
- portanto, $i \succ_{\nu(j)} j$ e $\nu(j) \succ_i \nu(i)$
  - ν não é estável

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

Prova:

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

Prova:

- $\pi = (\succ_{h_1}, \succ_{h_2}, \ldots, \succ_{h_n})$

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

Prova:

- $\pi = (\succ_{h_1}, \succ_{h_2}, \ldots, \succ_{h_n})$
- μ: emparelhamento encontrado pelo algoritmo com preferências masculinas π

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

Prova:

- $\pi = (\succ_{h_1}, \succ_{h_2}, \ldots, \succ_{h_n})$
- μ: emparelhamento encontrado pelo algoritmo com preferências masculinas π
- suponha que homem h_1 mente trocando $\succ_{h_1}$ por $\succ_*$

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

Prova:

- $\pi = (\succ_{h_1}, \succ_{h_2}, \ldots, \succ_{h_n})$
- μ: emparelhamento encontrado pelo algoritmo com preferências masculinas π
- suponha que homem h_1 mente trocando $\succ_{h_1}$ por $\succ_*$
- $\pi^1 = (\succ_*, \succ_{h_2}, \ldots, \succ_{h_n})$

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

Prova:

- $\pi = (\succ_{h_1}, \succ_{h_2}, \ldots, \succ_{h_n})$
- μ: emparelhamento encontrado pelo algoritmo com preferências masculinas π
- suponha que homem h_1 mente trocando $\succ_{h_1}$ por $\succ_*$
- $\pi^1 = (\succ_*, \succ_{h_2}, \ldots, \succ_{h_n})$
- ν: emparelhamento encontrado pelo algoritmo com preferências masculinas π^1

À prova de estratégia

Teorema. O algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens.

Prova:

- $\pi = (\succ_{h_1}, \succ_{h_2}, \ldots, \succ_{h_n})$
- μ: emparelhamento encontrado pelo algoritmo com preferências masculinas π
- suponha que homem h_1 mente trocando $\succ_{h_1}$ por $\succ_*$
- $\pi^1 = (\succ_*, \succ_{h_2}, \ldots, \succ_{h_n})$
- ν: emparelhamento encontrado pelo algoritmo com preferências masculinas π^1

Vamos mostrar que, se $\nu(h_1) \succ_{h_1} \mu(h_1)$ então ν não é estável

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$

$$\pi \to \mu \qquad \pi^1 \to \nu$$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$

$$\pi \to \mu \qquad \pi^1 \to \nu$$

Seja $R = \{h \colon \nu(h) \succ_h \mu(h)\}$ o conjunto de homens que melhoram

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$

$$\pi \to \mu \qquad \pi^1 \to \nu$$

Seja $R = \{h \colon \nu(h) \succ_h \mu(h)\}$ o conjunto de homens que melhoram

Vamos mostrar que para $h \in R$ onde $m = \nu(h)$ e $h' = \mu(m)$, h' pertence a R

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

Seja $R = \{h \colon \nu(h) \succ_h \mu(h)\}$ o conjunto de homens que melhoram

Vamos mostrar que para $h \in R$ onde $m = \nu(h)$ e $h' = \mu(m)$, h' pertence a R

- Se $h' = h_1$, nada a fazer (já supomos que $h_1 \in R$)

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

Seja $R = \{h \colon \nu(h) \succ_h \mu(h)\}$ o conjunto de homens que melhoram

Vamos mostrar que para $h \in R$ onde $m = \nu(h)$ e $h' = \mu(m)$, h' pertence a R

- Se $h' = h_1$, nada a fazer (já supomos que $h_1 \in R$)
- Como $m \succ_h \mu(h)$ a estabilidade de μ implica que $h' \succ_m h$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

Seja $R = \{h \colon \nu(h) \succ_h \mu(h)\}$ o conjunto de homens que melhoram

Vamos mostrar que para $h \in R$ onde $m = \nu(h)$ e $h' = \mu(m)$, h' pertence a R

- Se $h' = h_1$, nada a fazer (já supomos que $h_1 \in R$)
- Como $m \succ_h \mu(h)$ a estabilidade de μ implica que $h' \succ_m h$
- A estabilidade de ν para π^1 implica que $\nu(h') \succ_{h'} m$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

Seja $R = \{h \colon \nu(h) \succ_h \mu(h)\}$ o conjunto de homens que melhoram

Vamos mostrar que para $h \in R$ onde $m = \nu(h)$ e $h' = \mu(m)$, h' pertence a R

- Se $h' = h_1$, nada a fazer (já supomos que $h_1 \in R$)
- Como $m \succ_h \mu(h)$ a estabilidade de μ implica que $h' \succ_m h$
- A estabilidade de ν para π^1 implica que $\nu(h') \succ_{h'} m$
- Concluímos que $h' \in R$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

Seja $R = \{h \colon \nu(h) \succ_h \mu(h)\}$ o conjunto de homens que melhoram

Vamos mostrar que para $h \in R$ onde $m = \nu(h)$ e $h' = \mu(m)$, h' pertence a R

- Se $h' = h_1$, nada a fazer (já supomos que $h_1 \in R$)
- Como $m \succ_h \mu(h)$ a estabilidade de μ implica que $h' \succ_m h$
- A estabilidade de ν para π^1 implica que $\nu(h') \succ_{h'} m$
- Concluímos que $h' \in R$

Definimos $S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$

$$\pi \to \mu \qquad \pi^1 \to \nu$$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$

$$\pi \to \mu \qquad \pi^1 \to \nu$$

$$R = \{h \colon \nu(h) \succ_h \mu(h)\}$$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$

$$\pi \to \mu \qquad \pi^1 \to \nu$$

$$R = \{h \colon \nu(h) \succ_h \mu(h)\}$$

$$S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$

$$\pi \to \mu \qquad \pi^1 \to \nu$$

$R = \{h \colon \nu(h) \succ_h \mu(h)\}$

$S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$

$\mu(m) \succ_m \nu(m)$ para todo $m \in S$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

$R = \{h \colon \nu(h) \succ_h \mu(h)\}$

$S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$

$\mu(m) \succ_m \nu(m)$ para todo $m \in S$

Seja h o último homem em R a fazer uma proposta:

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

$R = \{h \colon \nu(h) \succ_h \mu(h)\}$

$S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$

$\mu(m) \succ_m \nu(m)$ para todo $m \in S$

Seja h o último homem em R a fazer uma proposta:

- proposta é feita para $m = \mu(h) \in S$ que rejeitou $\nu(m)$ em alguma iteração anterior

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

$R = \{h \colon \nu(h) \succ_h \mu(h)\}$

$S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$

$\mu(m) \succ_m \nu(m)$ para todo $m \in S$

Seja h o último homem em R a fazer uma proposta:

- proposta é feita para $m = \mu(h) \in S$ que rejeitou $\nu(m)$ em alguma iteração anterior
- quando h propõe para m, ela rejeita uma proposta de algum $h' \notin R$ tal que $h' \succ_m \nu(m)$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

$R = \{h \colon \nu(h) \succ_h \mu(h)\}$

$S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$

$\mu(m) \succ_m \nu(m)$ para todo $m \in S$

Seja h o último homem em R a fazer uma proposta:

- proposta é feita para $m = \mu(h) \in S$ que rejeitou $\nu(m)$ em alguma iteração anterior
- quando h propõe para m, ela rejeita uma proposta de algum $h' \notin R$ tal que $h' \succ_m \nu(m)$
- Como $h' \notin R$, temos que $m \succ_{h'} \mu(h') \succeq_{h'} \nu(h')$

À prova de estratégia

$$\pi = (\succ_{h_1}, \ldots) \qquad \pi^1 = (\succ_*, \ldots)$$
$$\pi \to \mu \qquad \pi^1 \to \nu$$

$R = \{h \colon \nu(h) \succ_h \mu(h)\}$

$S = \{m \colon \nu(h) \in R\} = \{m \colon \mu(h) \in R\}$

$\mu(m) \succ_m \nu(m)$ para todo $m \in S$

Seja h o último homem em R a fazer uma proposta:

- proposta é feita para $m = \mu(h) \in S$ que rejeitou $\nu(m)$ em alguma iteração anterior
- quando h propõe para m, ela rejeita uma proposta de algum $h' \notin R$ tal que $h' \succ_m \nu(m)$
- Como $h' \notin R$, temos que $m \succ_{h'} \mu(h') \succeq_{h'} \nu(h')$
- Como $h' \neq h_1$, (h', m) é um par bloqueador para ν em π^1

À prova de estratégia (?)

Vimos que o algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens

À prova de estratégia (?)

Vimos que o algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens

E para as mulheres?

À prova de estratégia (?)

Vimos que o algoritmo da aceitação postergada com proposta masculina é à prova de estratégia para os homens

E para as mulheres?

Voltamos ao exemplo...

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 :

m_2 :

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 :

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 : h_2

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 : h_2 h_3

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : h_2 h_3

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : ~~h_2~~ h_3

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : ~~h_2~~ h_3

m_2 : h_1

m_3 : h_2

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : ~~h_2~~ h_3

m_2 : h_1

m_3 : h_2

Emparelhamento produzido: $\{(h_1, m_2), (h_2, m_3), (h_3, m_1)\}$

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 :

m_2 :

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 :

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 : h_2

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 1:

m_1 : h_2 h_3

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : h_2 h_3

m_2 : h_1

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1 : h_2 h_3

m_2 : h_1

m_3 :

m_1 mente e rejeita h_3

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1: h_2 ~~h_3~~

m_2: h_1

m_3:

m_1 mente e rejeita h_3

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 2:

m_1: h_2 ~~h_3~~

m_2: h_1 h_3

m_3:

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 3:

m_1 : h_2 ~~h_3~~

m_2 : h_1 h_3

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 3:

m_1 : h_2 ~~h_3~~

m_2 : ~~h_1~~ h_3

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 3:

m_1 : h_2 ~~h_3~~ h_1

m_2 : ~~h_1~~ h_3

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 4:

m_1 : h_2 ~~h_3~~ h_1

m_2 : ~~h_1~~ h_3

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 4:

m_1 : ~~h_2~~ ~~h_3~~ h_1

m_2 : ~~h_1~~ h_3

m_3 :

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 4:

m_1: ~~h_2~~ ~~h_3~~ h_1

m_2: ~~h_1~~ h_3

m_3: h_2

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 4:

m_1: ~~h_2~~ ~~h_3~~ h_1

m_2: ~~h_1~~ h_3

m_3: h_2

Emparelhamento produzido: $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 4:

m_1: ~~h_2~~ ~~h_3~~ h_1

m_2: ~~h_1~~ h_3

m_3: h_2

Emparelhamento produzido: $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$

Emparelhamento anterior: $\{(h_1, m_2), (h_2, m_3), (h_3, m_1)\}$

Algoritmo no exemplo

$\succ_{h_1}$	$\succ_{h_2}$	$\succ_{h_3}$	$\succ_{m_1}$	$\succ_{m_2}$	$\succ_{m_3}$
m_2	m_1	m_1	h_1	h_3	h_1
m_1	m_3	m_2	h_3	h_1	h_3
m_3	m_2	m_3	h_2	h_2	h_2

Rodada 4:

m_1: ~~h_2~~ ~~h_3~~ h_1

m_2: ~~h_1~~ h_3

m_3: h_2

Emparelhamento produzido: $\{(h_1, m_1), (h_2, m_3), (h_3, m_2)\}$

Emparelhamento anterior: $\{(h_1, m_2), (h_2, m_3), (h_3, m_1)\}$

A mulher m_1 melhorou ao mentir

Problema das admissões em universidades

No problema das admissões em universidades:

Problema das admissões em universidades

No problema das admissões em universidades:

- Temos um conjunto $C = \{c_1, \ldots, c_n\}$ de universidades

Problema das admissões em universidades

No problema das admissões em universidades:

- Temos um conjunto $C = \{c_1, \ldots, c_n\}$ de universidades
  - Cada universidade c_i tem uma quota q_i

Problema das admissões em universidades

No problema das admissões em universidades:

- Temos um conjunto $C = \{c_1, \ldots, c_n\}$ de universidades
  - Cada universidade c_i tem uma quota q_i
- e temos um conjunto $S = \{s_1, \ldots, s_m\}$ de alunos

Problema das admissões em universidades

No problema das admissões em universidades:

- Temos um conjunto $C = \{c_1, \ldots, c_n\}$ de universidades
  - Cada universidade c_i tem uma quota q_i
- e temos um conjunto $S = \{s_1, \ldots, s_m\}$ de alunos
- Cada aluno tem uma ordem de preferência sobre as universidades

Problema das admissões em universidades

No problema das admissões em universidades:

- Temos um conjunto $C = \{c_1, \ldots, c_n\}$ de universidades
  - Cada universidade c_i tem uma quota q_i
- e temos um conjunto $S = \{s_1, \ldots, s_m\}$ de alunos
- Cada aluno tem uma ordem de preferência sobre as universidades
- Cada universidade c_i tem uma ordem de preferência sobre os subconjuntos de alunos de tamanho q_i

Problema da admissão em universidades

Roth'85: *The college admissions problem is not equivalent to the marriage problem*

Problema da admissão em universidades

Roth'85: *The college admissions problem is not equivalent to the marriage problem*

Para o problema do casamento estável vale que

Problema da admissão em universidades

Roth'85: *The college admissions problem is not equivalent to the marriage problem*

Para o problema do casamento estável vale que

- não existe alocação instável que todos os homens preferem a alocação ótima-masculina

Problema da admissão em universidades

Roth'85: *The college admissions problem is not equivalent to the marriage problem*

Para o problema do casamento estável vale que

- não existe alocação instável que todos os homens preferem a alocação ótima-masculina
- existe mecanismo à prova de estratégia para os homens

Problema da admissão em universidades

Roth'85: *The college admissions problem is not equivalent to the marriage problem*

Para o problema do casamento estável vale que

- não existe alocação instável que todos os homens preferem a alocação ótima-masculina
- existe mecanismo à prova de estratégia para os homens

Esses resultados não valem para o problema das admissões em universidades

Problema da admissão em universidades

Roth'85: *The college admissions problem is not equivalent to the marriage problem*

Para o problema do casamento estável vale que

- não existe alocação instável que todos os homens preferem a alocação ótima-masculina
- existe mecanismo à prova de estratégia para os homens

Esses resultados não valem para o problema das admissões em universidades

A dificuldade reside no fato que as universidades têm preferências sobre subconjuntos de alunos

Suplementar vs. Complementar

Se considerarmos que os alunos seguem uma suplementariedade:

Suplementar vs. Complementar

Se considerarmos que os alunos seguem uma suplementariedade:

- Isto é, as preferências das universidades não são sobre subconjuntos

Suplementar vs. Complementar

Se considerarmos que os alunos seguem uma suplementariedade:

- Isto é, as preferências das universidades não são sobre subconjuntos
- mas sim de uma preferência sobre os alunos

Suplementar vs. Complementar

Se considerarmos que os alunos seguem uma suplementariedade:

- Isto é, as preferências das universidades não são sobre subconjuntos
- mas sim de uma preferência sobre os alunos

Então, o problema é equivalente ao problema do casamento estável

Suplementar vs. Complementar

Se considerarmos que os alunos seguem uma suplementariedade:

- Isto é, as preferências das universidades não são sobre subconjuntos
- mas sim de uma preferência sobre os alunos

Então, o problema é equivalente ao problema do casamento estável

- basta dividir uma universidade com cota q_i

Suplementar vs. Complementar

Se considerarmos que os alunos seguem uma suplementariedade:

- Isto é, as preferências das universidades não são sobre subconjuntos
- mas sim de uma preferência sobre os alunos

Então, o problema é equivalente ao problema do casamento estável

- basta dividir uma universidade com cota q_i
- em q_i universidades de cota 1

Suplementar vs. Complementar

Porém, existem situações onde complementariedade existe:

Suplementar vs. Complementar

Porém, existem situações onde complementariedade existe:

- Casais de médicos em programas de residência

Suplementar vs. Complementar

Porém, existem situações onde complementariedade existe:

- Casais de médicos em programas de residência
- Casais de crianças em programas de adoção

Suplementar vs. Complementar

Porém, existem situações onde complementariedade existe:

- Casais de médicos em programas de residência
- Casais de crianças em programas de adoção

Ou seja, nem sempre é possível usar o algoritmo da proposta postergada