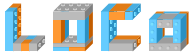
Universidade de Campinas (UNICAMP)
Instituto de Computação (IC)



Algoritmo de aproximação assintoticamente ótimo para o problema do alugador de veículos

Lehilton L. C. Pedrosa Greis Y. O. Quesquén Rafael C. S. Schouery

Dia 78 da quarentena (ou 28/5/2020)
Seminários online de Grafos, Algoritmos e Combinatória

Relembrando o TSP clássico

Relembrando o TSP clássico

Um *caixeiro viajante*

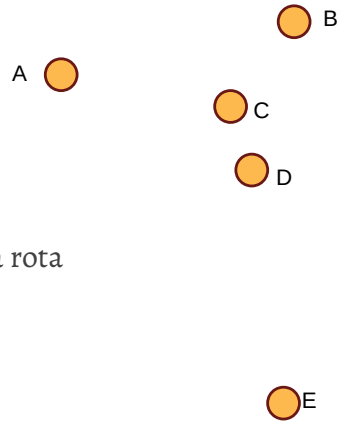
- tem um conjunto de cidades a visitar
- decide alugar um veículo para fazer isso

Relembrando o TSP clássico

Um *caixeiro viajante*

- tem um conjunto de cidades a visitar
- decide alugar um veículo para fazer isso

  Objetivo: minimizar o comprimento total da rota

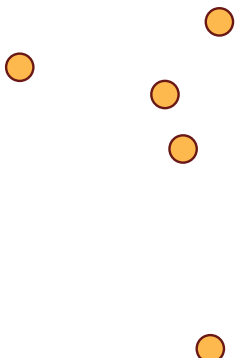


Formalizando TSP

- **Entrada:** Um grafo completo G e uma distância d nas arestas
- **Objetivo:** Encontrar uma rota C de G que minimiza $d(C)$

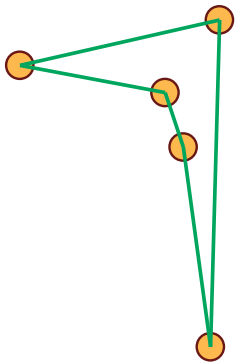
Formalizando TSP

- **Entrada:** Um grafo completo G e uma distância d nas arestas
- **Objetivo:** Encontrar uma rota C de G que minimiza $d(C)$



Formalizando TSP

- **Entrada:** Um grafo completo G e uma distância d nas arestas
- **Objetivo:** Encontrar uma rota C de G que minimiza $d(C)$



TSP é difícil

Ciclo hamiltoniano

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?
- sei lá, esse problema é NP-difícil!

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?
- sei lá, esse problema é NP-difícil!

Assim como TSP

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?
- sei lá, esse problema é NP-difícil!

Assim como TSP

1. defina $d(u, v) = \Big\{$

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?
- sei lá, esse problema é NP-difícil!

Assim como TSP

1. defina $d(u, v) = \begin{cases} 0 & \text{se } (u, v) \text{ é aresta de } G, \end{cases}$

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?
- sei lá, esse problema é NP-difícil!

Assim como TSP

1. defina $d(u, v) = \begin{cases} 0 & \text{se } (u, v) \text{ é aresta de } G, \\ 1 & \text{se não for} \end{cases}$

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?
- sei lá, esse problema é NP-difícil!

Assim como TSP

1. defina $d(u, v) = \begin{cases} 0 & \text{se } (u, v) \text{ é aresta de } G, \\ 1 & \text{se não for} \end{cases}$
2. resolva essa instância de TSP

TSP é difícil

Ciclo hamiltoniano

- **Pergunta:** dado um grafo G, há um ciclo que percorre todos vértices?
- sei lá, esse problema é NP-difícil!

Assim como TSP

1. defina $d(u, v) = \begin{cases} 0 & \text{se } (u, v) \text{ é aresta de } G, \\ 1 & \text{se não for} \end{cases}$
2. resolva essa instância de TSP
3. há um ciclo hamiltoniano sse TSP tem valor ótimo 0

TSP é difícil de aproximar

Um algoritmo de aproximação

TSP é difícil de aproximar

Um algoritmo de aproximação

1. executa em tempo polinomial

TSP é difícil de aproximar

Um algoritmo de aproximação

1. executa em tempo polinomial
2. encontra uma solução com valor $\text{SOL} \leq \alpha \cdot \text{OPT}$

TSP é difícil de aproximar

Um algoritmo de aproximação

1. executa em tempo polinomial
2. encontra uma solução com valor $\text{SOL} \leq \alpha \cdot \text{OPT}$

Também não vou aproximar TSP

TSP é difícil de aproximar

Um algoritmo de aproximação

1. executa em tempo polinomial
2. encontra uma solução com valor $\text{SOL} \leq \alpha \cdot \text{OPT}$

Também não vou aproximar TSP

- uma solução teria custo $\alpha \cdot \text{OPT} = 0$ sse existe ciclo hamiltoniano

TSP é difícil de aproximar

Um algoritmo de aproximação

1. executa em tempo polinomial
2. encontra uma solução com valor $\text{SOL} \leq \alpha \cdot \text{OPT}$

Também não vou aproximar TSP

- uma solução teria custo $\alpha \cdot \text{OPT} = 0$ sse existe ciclo hamiltoniano
- se existisse α-aproximação...

TSP é difícil de aproximar

Um algoritmo de aproximação

1. executa em tempo polinomial
2. encontra uma solução com valor $\text{SOL} \leq \alpha \cdot \text{OPT}$

Também não vou aproximar TSP

- uma solução teria custo $\alpha \cdot \text{OPT} = 0$ sse existe ciclo hamiltoniano
- se existisse α-aproximação...
- ...eu saberia responder ciclo hamiltoniano

Entra métrica na jogada

Entra métrica na jogada

A função de distância é boba

Entra métrica na jogada

A função de distância é boba

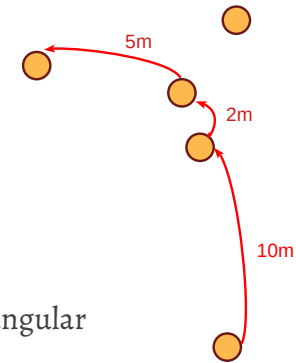
- tomar atalhos deveriam ajudar

Entra métrica na jogada

A função de distância é boba

- tomar atalhos deveriam ajudar
- mas naquela distância d, não temos desigualdade triangular

Entra métrica na jogada



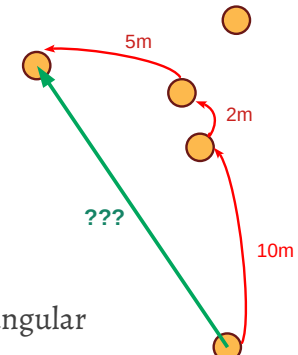
A função de distância é boba

- tomar atalhos deveriam ajudar
- mas naquela distância d, não temos desigualdade triangular

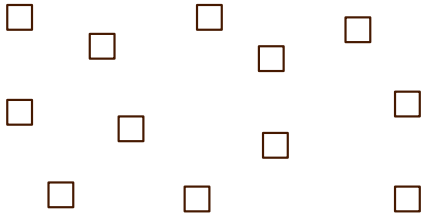
Entra métrica na jogada

A função de distância é boba

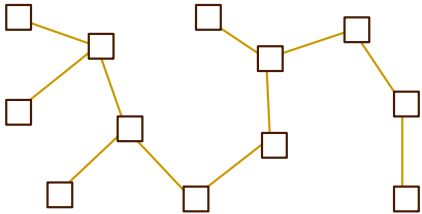
- tomar atalhos deveriam ajudar
- mas naquela distância d, não temos desigualdade triangular



Aproximando o problema

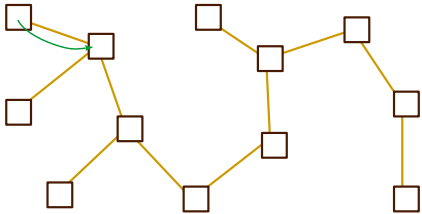


Aproximando o problema



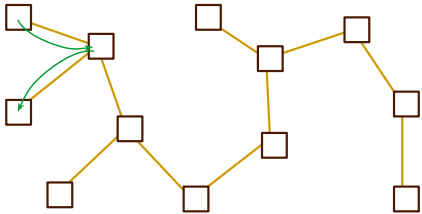
1. Construa uma $\mathcal{A}.\mathcal{G}.\mathcal{M}$.

Aproximando o problema



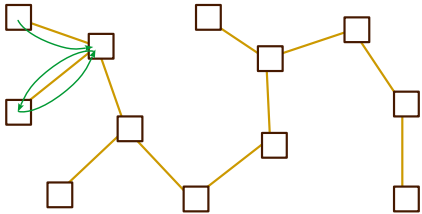
1. Construa uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$
2. Percorra a árvore em profundidade

Aproximando o problema



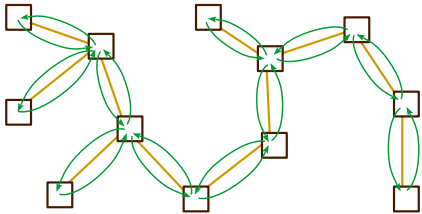
1. Construa uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$
2. Percorra a árvore em profundidade

Aproximando o problema



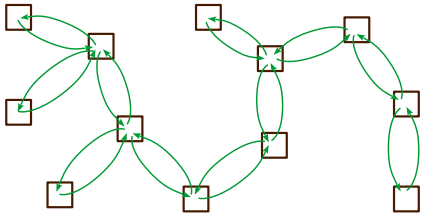
1. Construa uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$
2. Percorra a árvore em profundidade

Aproximando o problema



1. Construa uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$
2. Percorra a árvore em profundidade

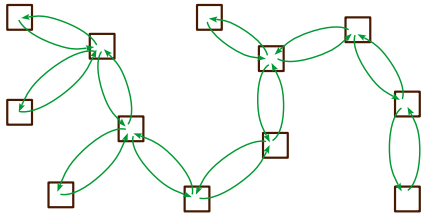
Aproximando o problema



1. Construa uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$
2. Percorra a árvore em profundidade
3. Devolva as arestas duplicadas

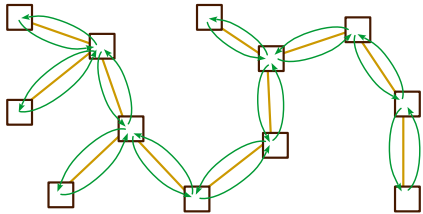
Analisando a solução

Qual o valor SOL da solução obtida?



Analisando a solução

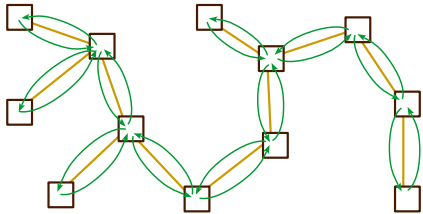
Qual o valor SOL da solução obtida?



$$\text{SOL} \leq 2\text{AGM}$$

Analisando a solução

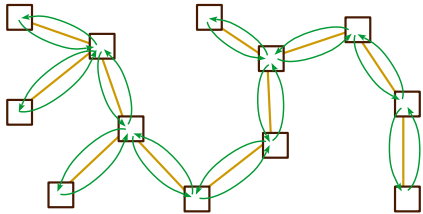
Qual o valor SOL da solução obtida?



$$\text{SOL} \leq 2\text{AGM} \leq 2\text{OPT}$$

Analisando a solução

Qual o valor SOL da solução obtida?



$$\text{SOL} \leq 2\text{AGM} \leq 2\text{OPT}$$

Isso é, obtivemos uma 2-aproximação.

Um cenário atualizado

Hoje, podemos escolher entre diversas possibilidades

Um cenário atualizado

Hoje, podemos escolher entre diversas possibilidades

- um turista usa táxi, apps, etc.
- um caixeiro pode alugar diversos veículos
- cada veículo percorre partes diferentes da rota

Um cenário atualizado

Hoje, podemos escolher entre diversas possibilidades

- um turista usa táxi, apps, etc.
- um caixeiro pode alugar diversos veículos
- cada veículo percorre partes diferentes da rota

  Objtivo: Determinar partes e minimizar o custo total

Um cenário atualizado

Hoje, podemos escolher entre diversas possibilidades

- um turista usa táxi, apps, etc.
- um caixeiro pode alugar diversos veículos
- cada veículo percorre partes diferentes da rota

  Objtivo: Determinar partes e minimizar o custo total

Problema do Alugador de Veículos (CARS)

Um cenário atualizado

Hoje, podemos escolher entre diversas possibilidades

- um turista usa táxi, apps, etc.
- um caixeiro pode alugar diversos veículos
- cada veículo percorre partes diferentes da rota

  Objtivo: Determinar partes e minimizar o custo total

Problema do Alugador de Veículos (CARS)

- o custo de percorrer cada parte depende da empresa
- há uma taxa de devolução para cada carro alugado

Generalizando TSP: UCARS

Um pouco mais difícil de descrever que TSP

Generalizando TSP: UCARS

Um pouco mais difícil de descrever que TSP

- **Entrada:**
  - grafo completo G
  - r funções de distância em G: $d_1, d_2, \ldots, d_r$
  - taxa de retorno $g \geq 0$

Generalizando TSP: UCARS

Um pouco mais difícil de descrever que TSP

- **Entrada:**
  - grafo completo G
  - r funções de distância em G: $d_1, d_2, \dots, d_r$
  - taxa de retorno $g \geq 0$
- **Objetivo:**
  - Encontrar passeios $S_1, S_2, \dots, S_s$ que percorrem todas cidades e minimizam

$$\sum_{S_j} [d_j(S_j) + g]$$

Generalizando TSP: UCARS

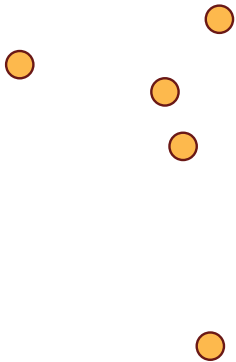
Um pouco mais difícil de descrever que TSP

- **Entrada:**
  - grafo completo G
  - r funções de distância em G: $d_1, d_2, \dots, d_r$
  - taxa de retorno $g \geq 0$
- **Objetivo:**
  - Encontrar passeios $S_1, S_2, \dots, S_s$ que percorrem todas cidades e minimizam

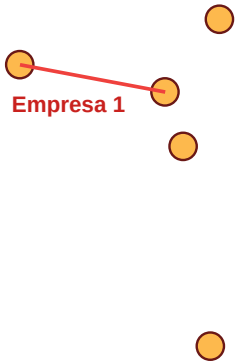
$$\sum_{S_j} [d_j(S_j) + g]$$

Chamamos de CARS Uniforme (UCARS) porque g é fixo

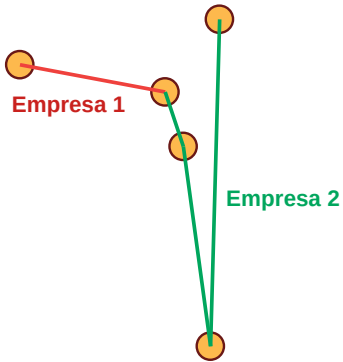
Exemplo



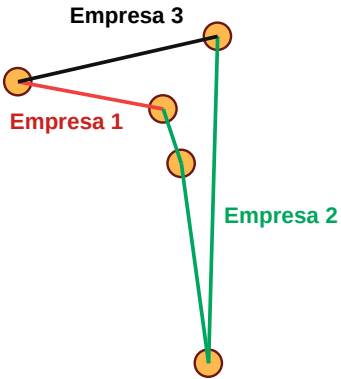
Exemplo



Exemplo



Exemplo



Uma rápida revisão da literatura

TSP

- estudado principalmente num espaço métrico
- duplicar as arestas de uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$ de G é uma 2-aproximação
- melhor fator é $3/2$ de Christofides (em 1976!)

Uma rápida revisão da literatura

TSP

- estudado principalmente num espaço métrico
- duplicar as arestas de uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$ de G é uma 2-aproximação
- melhor fator é $3/2$ de Christofides (em 1976!)

CARS

- introduzido por Goldbarg et al. (2012)
- tratado usando muitas heuristics (algoritmos genéticos, etc.)
- e formulações matemáticas (PLI, fluxo em rede, etc.)
- nenhum estudo em aproximação

Uma rápida revisão da literatura

TSP

- estudado principalmente num espaço métrico
- duplicar as arestas de uma $\mathcal{A}.\mathcal{G}.\mathcal{M}.$ de G é uma 2-aproximação
- melhor fator é $3/2$ de Christofides (em 1976!)

CARS

- introduzido por Goldbarg et al. (2012)
- tratado usando muitas heuristics (algoritmos genéticos, etc.)
- e formulações matemáticas (PLI, fluxo em rede, etc.)
- nenhum estudo em aproximação

Pergunta: Existe alguma aproximação?

Nossas tarefas

Inaproximabilidade:

- nenhuma aproximação existe se queremos visitar cada cidade só uma vez
- UCARS não tem $c \log n$-aproximação com $c < 1$

Nossas tarefas

Inaproximabilidade:

- nenhuma aproximação existe se queremos visitar cada cidade só uma vez
- UCARS não tem $c \log n$-aproximação com $c < 1$

Aproximação:

- redução para o *Group TSP*
- uma $O(\log n)$-aproximação

Nenhuma aproximação em geral

Se uma solução dever ser um ciclo (sem repetir vértices)

Nenhuma aproximação em geral

Se uma solução dever ser um ciclo (sem repetir vértices)

- aproximar UCARS resolveria ciclo hamiltoniano eficientemente

Nenhuma aproximação em geral

Se uma solução dever ser um ciclo (sem repetir vértices)

- aproximar UCARS resolveria ciclo hamiltoniano eficientemente
- então não existe aproximação (hum… se $P \neq NP$)

Nenhuma aproximação em geral

Se uma solução dever ser um ciclo (sem repetir vértices)

- aproximar UCARS resolveria ciclo hamiltoniano eficientemente
- então não existe aproximação (hum… se $P \neq NP$)

Mas para nossa motivação

Nenhuma aproximação em geral

Se uma solução dever ser um ciclo (sem repetir vértices)

- aproximar UCARS resolveria ciclo hamiltoniano eficientemente
- então não existe aproximação (hum… se $P \neq NP$)

Mas para nossa motivação

- Uma solução é qualquer rota geradora

Nenhuma aproximação em geral

Se uma solução dever ser um ciclo (sem repetir vértices)

- aproximar UCARS resolveria ciclo hamiltoniano eficientemente
- então não existe aproximação (hum… se $P \neq NP$)

Mas para nossa motivação

- Uma solução é qualquer rota geradora
- Permitimos repetição

Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.

Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

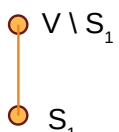
- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

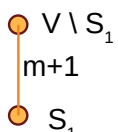


Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

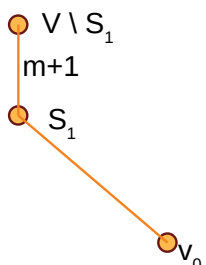


Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

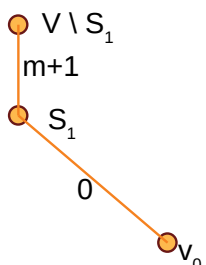


Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

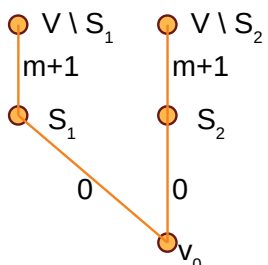


Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

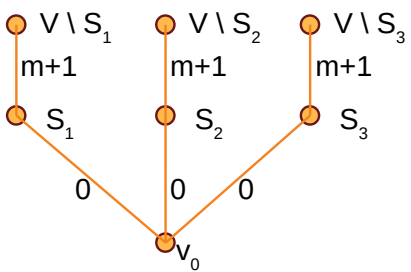


Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:

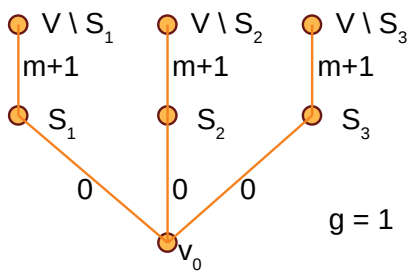


Inaproximabilidade

Teorema: Não há $c \log n$-aproximação (a não ser que P $=$ NP).

Reduzimos cobertura por conjuntos para UCARS.

- Dados conjuntos $S_1, S_2, \ldots, S_m \subset U$, selecionar k deles para cobrir U.
- Criamos uma instância correspondente de UCARS:



Correspondência com Group TSP

Group TSP: Encontrar uma rota que visita cada grupo pelo menos uma vez

Correspondência com Group TSP

Group TSP: Encontrar uma rota que visita cada grupo pelo menos uma vez

Observação: UCARS pode ser reduzido a Group TSP.

Correspondência com Group TSP

Group TSP: Encontrar uma rota que visita cada grupo pelo menos uma vez

Observação: UCARS pode ser reduzido a Group TSP.

Ideia:

1. Faça r cópias de G (uma para cada empresa)
2. Ligue vértices correspondentes a uma cidade com curso g
3. Cada cidade corresponde a um grupo

Correspondência com Group TSP

Group TSP: Encontrar uma rota que visita cada grupo pelo menos uma vez

Observação: UCARS pode ser reduzido a Group TSP.

Ideia:

1. Faça r cópias de G (uma para cada empresa)
2. Ligue vértices correspondentes a uma cidade com curso g
3. Cada cidade corresponde a um grupo

Consequências:

- existe uma $O(\log^2 n \log r)$-aproximação
- o fator não é bom, mas a ideia generaliza a variantes
- mas muito ruim para UCARS, que tem limitante $\Omega(\log n)$

Um algoritmo melhorado

Observe: cada trecho da solução é um conjunto de cidades

Um algoritmo melhorado

Observe: cada trecho da solução é um conjunto de cidades

Estratégia central:

Um algoritmo melhorado

Observe: cada trecho da solução é um conjunto de cidades

Estratégia central:

- enumerar componentes S para cada empresa j:

Um algoritmo melhorado

Observe: cada trecho da solução é um conjunto de cidades

Estratégia central:

- enumerar componentes S para cada empresa j: $\mathscr{S} = \{(j, S) : S \subseteq V, j\}$

Um algoritmo melhorado

Observe: cada trecho da solução é um conjunto de cidades

Estratégia central:

- enumerar componentes S para cada empresa j: $\mathscr{S} = \{(j, S) : S \subseteq V, j\}$
- queremos selecionar componentes

Um algoritmo melhorado

Observe: cada trecho da solução é um conjunto de cidades

Estratégia central:

- enumerar componentes S para cada empresa j: $\mathscr{S} = \{(j, S) : S \subseteq V, j\}$
- queremos selecionar componentes
- que cubram todas cidades

Um algoritmo melhorado

Observe: cada trecho da solução é um conjunto de cidades

Estratégia central:

- enumerar componentes S para cada empresa j: $\mathscr{S} = \{(j, S) : S \subseteq V, j\}$
- queremos selecionar componentes
- que cubram todas cidades

$\Rightarrow$ enunciamos o problema como uma *cobertura por conjuntos*!

Formulação PLI

$$
\begin{aligned}
\text{minimizar} \quad & \sum_{(j,S)} x_{(j,S)} (\text{MST}_j(S) + g) \\
\text{sujeito a} \quad & \textstyle\sum_{(j,S):v \in S} x_{(j,S)} \geq 1 \quad \forall\ v \in V, \\
& x_{(j,S)} \in \{0,1\} \qquad\quad \forall\ (j,S) \in \mathscr{S}.
\end{aligned}
\tag{IP}
$$

Formulação PLI

$$
\begin{aligned}
\text{minimizar} \quad & \sum_{(j,S)} x_{(j,S)}(\text{MST}_j(S) + g) \\
\text{sujeito a} \quad & \textstyle\sum_{(j,S):v\in S} x_{(j,S)} \geq 1 \quad \forall\ v \in V, \\
& x_{(j,S)} \in \{0,1\} \qquad\quad \forall\ (j,S) \in \mathscr{S}.
\end{aligned}
\tag{IP}
$$

- reescrevemos como um programa linear inteiro
- mesmo com restrições de integralidade, ela é uma relaxação

Formulação PLI

$$
\begin{array}{lll}
\text{minimizar} & \displaystyle\sum_{(j,S)} x_{(j,S)}(\text{MST}_j(S) + g) & \\
\text{sujeito a} & \sum_{(j,S):v\in S} x_{(j,S)} \geq 1 & \forall\ v \in V, \\
& x_{(j,S)} \in \{0,1\} & \forall\ (j,S) \in \mathscr{S}.
\end{array}
\qquad \text{(IP)}
$$

- reescrevemos como um programa linear inteiro
- mesmo com restrições de integralidade, ela é uma relaxação
- suponha que temos uma solução fracionária x do PL
- queremos convertê-la em uma solução integral

Arredondando a solução do PL

- recebemos a solução fracionária x do PL

Arredondando a solução do PL

- recebemos a solução fracionária x do PL
- executamos três fases

Arredondando a solução do PL

- recebemos a solução fracionária x do PL
- executamos três fases
  A) Fase de cobertura
    - selecione cada componente (i, S) com probabilidade $(\log n) \cdot x_{(i,S)}$
    - cada cidade é coberta com *alta probabilidade*

Arredondando a solução do PL

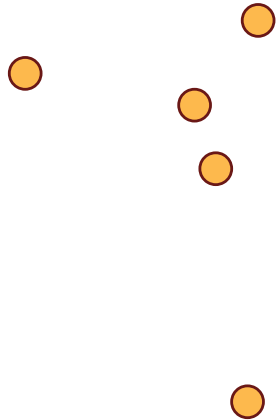
- recebemos a solução fracionária x do PL
- executamos três fases
  A) Fase de cobertura
     - selecione cada componente (i, S) com probabilidade $(\log n) \cdot x_{(i,S)}$
     - cada cidade é coberta com *alta probabilidade*
  B) Fase de conexão
     - garanta que as componentes sejam conectadas

Arredondando a solução do PL

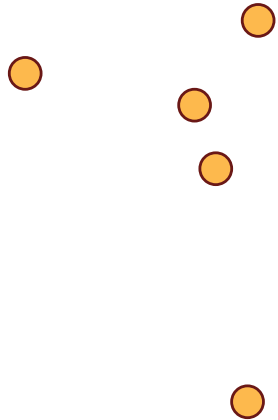
- recebemos a solução fracionária x do PL
- executamos três fases
  - A) Fase de cobertura
    - selecione cada componente (i, S) com probabilidade $(\log n) \cdot x_{(i,S)}$
    - cada cidade é coberta com *alta probabilidade*
  - B) Fase de conexão
    - garanta que as componentes sejam conectadas
  - C) Fase de roteamento
    - Transforme as componentes em uma rota

Exemplo

Visão geral:



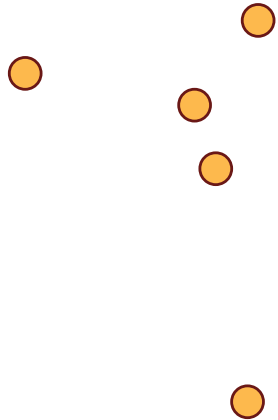
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.

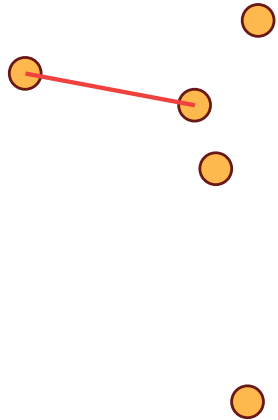
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.

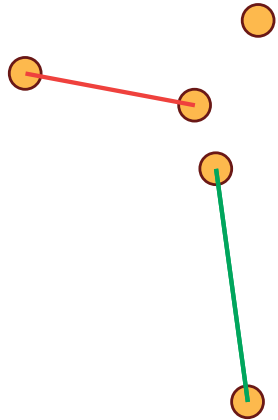
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.

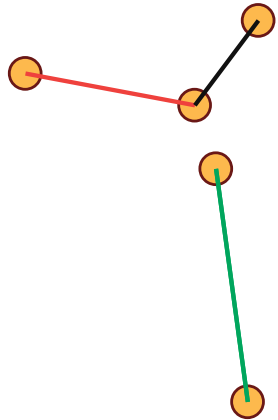
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.

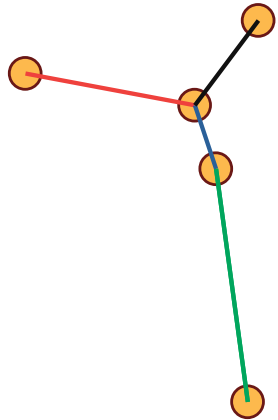
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.

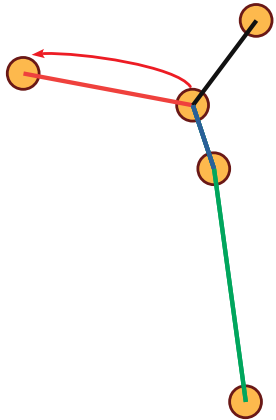
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.

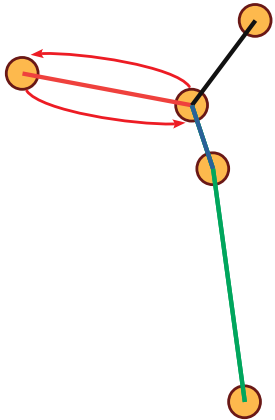
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

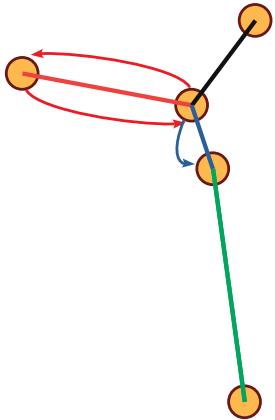
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

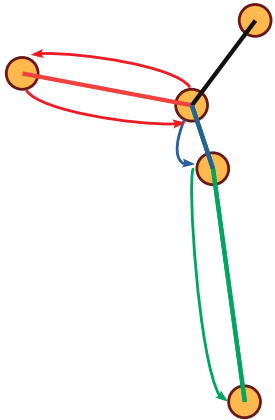
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

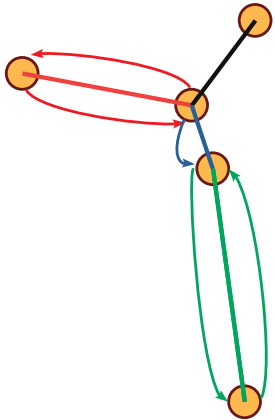
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

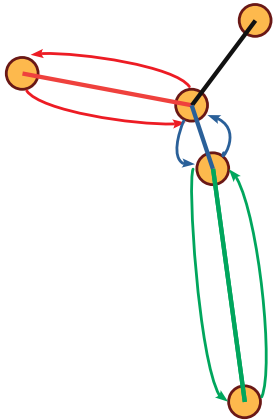
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

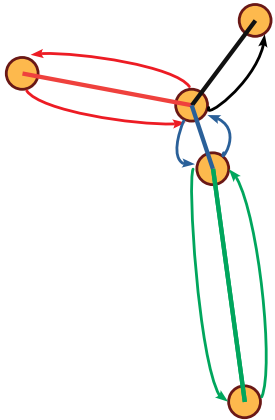
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

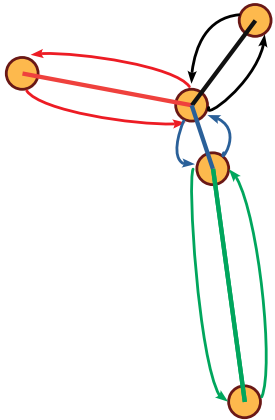
Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

Exemplo



Visão geral:

1. Encontre uma solução da relaxação.
2. Selecione componentes probabilisticamente.
3. Ligue componentes desconectadas.
4. Construa uma rota.

Análise do algoritmo de arredondando

O valor da solução é limitado pelas componentes selecionadas S aleatoriamente:

$$E\left[\sum_{(j,S)\in S}(\text{MST}_j(S)+g)\right]$$

.

Análise do algoritmo de arredondando

O valor da solução é limitado pelas componentes selecionadas S aleatoriamente:

$$E\left[\sum_{(j,S)\in S}(\text{MST}_j(S)+g)\right] = O(\log n)\sum_{(j,S)\in\mathscr{S}} x_{(j,S)}(\text{MST}_j(S)+g)$$

.

Análise do algoritmo de arredondando

O valor da solução é limitado pelas componentes selecionadas S aleatoriamente:

$$
\begin{aligned}
E\left[\sum_{(j,S)\in S}(\text{MST}_j(S)+g)\right] &= O(\log n)\sum_{(j,S)\in\mathscr{S}} x_{(j,S)}(\text{MST}_j(S)+g) \\
&= O(\log n)\text{OPT}(\text{IP}).
\end{aligned}
$$

Mas...

Ainda falta encontrar um solução do PL

Mas...

Ainda falta encontrar um solução do PL

- a formulação tem número exponencial de variáveis
- podemos tentar usar o método dos elipsóides

Mas...

Ainda falta encontrar um solução do PL

- a formulação tem número exponencial de variáveis
- podemos tentar usar o método dos elipsóides

Em seguida

- vamos encontrar uma solução x fracionária em tempo polinomial
- cujo valor não seja maior do que $O(\mathrm{OPT}(\mathrm{IP}))$

O método dos elipsóides

Teorema: Se o problema da separação é polinomial, então a relaxação do PLI é polinomial.

O método dos elipsóides

Teorema: Se o problema da separação é polinomial, então a relaxação do PLI é polinomial.

Problema da separação em geral:

O método dos elipsóides

Teorema: Se o problema da separação é polinomial, então a relaxação do PLI é polinomial.

Problema da separação em geral:

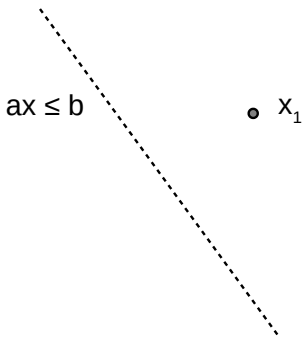
- dada uma solução inviável x_1

O método dos elipsóides

Teorema: Se o problema da separação é polinomial, então a relaxação do PLI é polinomial.

Problema da separação em geral:

- dada uma solução inviável x_1
- encontrar vetor a e número b tais que:

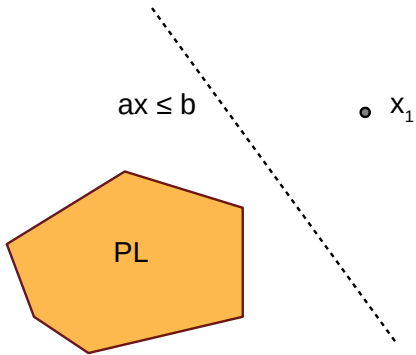


O método dos elipsóides

Teorema: Se o problema da separação é polinomial, então a relaxação do PLI é polinomial.

Problema da separação em geral:

- dada uma solução inviável x_1
- encontrar vetor a e número b tais que:
  1. $ax \leq b$ é desigualdade válida

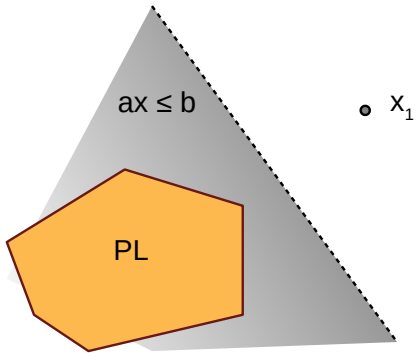


O método dos elipsóides

Teorema: Se o problema da separação é polinomial, então a relaxação do PLI é polinomial.

Problema da separação em geral:

- dada uma solução inviável x_1
- encontrar vetor a e número b tais que:
  1. $ax \leq b$ é desigualdade válida
  2. $ax_1 > b$



Dual da relaxação

$$
\begin{aligned}
\text{maximizar} \quad & \sum_{v \in V} y_v & & \\
\text{sujeito a} \quad & \sum_{v \in S} y_v - \text{MST}_i(S) \leq g & \forall\ (i, S) \in \mathscr{S} & \qquad \text{(D)} \\
& y_v \geq 0 & \forall\ v \in V. &
\end{aligned}
$$

O problema da separação

- a restrição do dual é

$$\sum_{v \in S} y_v - \text{MST}_i(S) \leq g$$

O problema da separação

- a restrição do dual é

$$\sum_{v \in S} y_v - \text{MST}_i(S) \leq g$$

- o problema da separação é o mesmo que maximizar o lado esquerdo

O problema da separação

- a restrição do dual é

$$\sum_{v \in S} y_v - \text{MST}_i(S) \leq g$$

- o problema da separação é o mesmo que maximizar o lado esquerdo
- esse é o Net Worth Maximization Problem (NWMP):

O problema da separação

- a restrição do dual é

$$\sum_{v \in S} y_v - \text{MST}_i(S) \leq g$$

- o problema da separação é o mesmo que maximizar o lado esquerdo
- esse é o Net Worth Maximization Problem (NWMP):
  - **Entrada:** grafo G, peso nas arestas w, peso nos vértices y

O problema da separação

- a restrição do dual é

$$\sum_{v \in S} y_v - \text{MST}_i(S) \leq g$$

- o problema da separação é o mesmo que maximizar o lado esquerdo
- esse é o Net Worth Maximization Problem (NWMP):
  - **Entrada:** grafo G, peso nas arestas w, peso nos vértices y
  - **Solução:** uma árvore $T \subseteq G$

O problema da separação

- a restrição do dual é

$$\sum_{v \in S} y_v - \text{MST}_i(S) \leq g$$

- o problema da separação é o mesmo que maximizar o lado esquerdo
- esse é o Net Worth Maximization Problem (NWMP):
  - **Entrada:** grafo G, peso nas arestas w, peso nos vértices y
  - **Solução:** uma árvore $T \subseteq G$
  - **Objetivo:** maximizar

$$\sum_{v \in V(T)} y_v - \sum_{e \in E(T)} w_e$$

Separar é (muito) difícil

Não podemos separar eficientemente

Separar é (muito) difícil

Não podemos separar eficientemente

- NWMP é NP-difícil

Separar é (muito) difícil

Não podemos separar eficientemente

- NWMP é NP-difícil
- mesmo encontrar uma aproximação constante é difícil!

Separar é (muito) difícil

Não podemos separar eficientemente

- NWMP é NP-difícil
- mesmo encontrar uma aproximação constante é difícil!

Estratégia alternativa

Separar é (muito) difícil

Não podemos separar eficientemente

- NWMP é NP-difícil
- mesmo encontrar uma aproximação constante é difícil!

Estratégia alternativa

- resolver uma modificação do dual

Separar é (muito) difícil

Não podemos separar eficientemente

- NWMP é NP-difícil
- mesmo encontrar uma aproximação constante é difícil!

Estratégia alternativa

- resolver uma modificação do dual
- mostrar que o PL correspondente tem valor $O(\mathrm{OPT}(\mathrm{IP}))$.

Modificando uma solução ótima

Ideia: distinguir componentes baratas de caras

- uma componente (j, S) é cara se $\text{MST}_j(S) \geq g$
- quebrar componentes caras em componentes baratas
- pagamos taxas de retorno adicionais
- mas o valor é pequeno comparado ao custo do percurso

Relaxação do PLI original

$$
\begin{aligned}
\text{minimizar} \quad & \sum_{(j,S)} x_{(j,S)}(\text{MST}_j(S) + g) \\
\text{sujeito a} \quad & \textstyle\sum_{(j,S):v\in S} x_{(j,S)} \geq 1 \quad \forall\ v \in V, \\
& x_{(j,S)} \geq 0 \qquad\qquad\quad \forall\ (j,S) \in \mathscr{S}
\end{aligned}
\tag{P}
$$

- Essa relaxação contém variáveis para todas as componentes

Relaxação do PLI modificada

$$
\begin{aligned}
\text{minimizar} \quad & \sum_{(j,S)} x_{(j,S)}(\text{MST}_j(S) + g) \\
\text{sujeito a} \quad & \textstyle\sum_{(j,S):v\in S} x_{(j,S)} \geq 1 \quad \forall\ v \in V, \\
& x_{(j,S)} \geq 0 \qquad\qquad\quad \forall\ (j,S) \in \mathscr{S}_{\leq g}
\end{aligned}
\tag{$\text{P}_{\leq g}$}
$$

- A modificação contém variáveis apenas para componentes baratas

Relaxação do PLI modificada

$$
\begin{aligned}
\text{minimizar} \quad & \sum_{(j,S)} x_{(j,S)}(\text{MST}_j(S) + g) \\
\text{sujeito a} \quad & \textstyle\sum_{(j,S):v\in S} x_{(j,S)} \geq 1 \quad \forall\ v \in V, \\
& x_{(j,S)} \geq 0 \qquad\qquad\quad \forall\ (j,S) \in \mathscr{S}_{\leq g}
\end{aligned}
\tag{$\text{P}_{\leq g}$}
$$

- A modificação contém variáveis apenas para componentes baratas
- Uma rota ótima pode não induzir uma solução viável dessa restrição
- Mas a solução modificada custa menos que quatro vezes mais:

$$\text{OPT}(\text{P}) \leq 4 \cdot \text{OPT}(\text{IP})$$

Dual da modificação

$$
\begin{aligned}
\text{maximizar} \quad & \sum_{v \in V} y_v & & \\
\text{sujeito a} \quad & \sum_{v \in S} y_v - \mathrm{MST}_i(S) \leq g & \forall\ (i, S) \in \mathscr{S}_{\leq g}, & \qquad (\mathrm{D}_{\leq g}) \\
& y_v \geq 0 & \forall\ v \in V. &
\end{aligned}
$$

Temos um outro problema de separação

Dual da modificação

$$
\begin{aligned}
\text{maximizar} \quad & \sum_{v \in V} y_v & & \\
\text{sujeito a} \quad & \sum_{v \in S} y_v - \mathrm{MST}_i(S) \leq g & \forall\ (i, S) \in \mathscr{S}_{\leq g}, & \qquad (\mathrm{D}_{\leq g}) \\
& y_v \geq 0 & \forall\ v \in V. &
\end{aligned}
$$

Temos um outro problema de separação

- é uma versão de NWMP ligeiramente diferente

Dual da modificação

$$
\begin{aligned}
\text{maximizar} \quad & \sum_{v \in V} y_v & & \\
\text{sujeito a} \quad & \sum_{v \in S} y_v - \text{MST}_i(S) \leq g & \forall\ (i, S) \in \mathscr{S}_{\leq g}, & \qquad (\text{D}_{\leq g}) \\
& y_v \geq 0 & \forall\ v \in V. &
\end{aligned}
$$

Temos um outro problema de separação

- é uma versão de NWMP ligeiramente diferente
- mas ainda muito complicada, melhor simplificar o PL

Dual da modificação restrito

$$
\begin{aligned}
&\text{maximizar} && \sum_{v \in V} y_v && && \\
&\text{sujeito a} && \sum_{v \in S} y_v && \leq g \quad \forall\ (i,S) \in \mathscr{S}_{\leq g}, && (\mathrm{D}'_{\leq g}) \\
& && y_v \geq 0 && \forall\ v \in V. &&
\end{aligned}
$$

As coisas estão ficando estranhas...

Dual da modificação restrito

$$
\begin{aligned}
\text{maximizar} \quad & \sum_{v \in V} y_v & & & & \\
\text{sujeito a} \quad & \sum_{v \in S} y_v & & \leq g \quad & \forall\ (i, S) \in \mathscr{S}_{\leq g}, \qquad & (\mathrm{D}'_{\leq g}) \\
& y_v \geq 0 & & & \forall\ v \in V. &
\end{aligned}
$$

As coisas estão ficando estranhas...

- separação mais simples: encontrar árvore T de custo g e com maior peso
- é o Budget Steiner Tree Problem (BSTP) e é NP-difícil. Oh Deus!

Dual da modificação restrito

$$
\begin{aligned}
\text{maximizar} \quad & \sum_{v \in V} y_v & & & & \\
\text{sujeito a} \quad & \sum_{v \in S} y_v & & \leq g \quad & \forall\ (i, S) \in \mathscr{S}_{\leq g}, \qquad & (\mathrm{D}'_{\leq g}) \\
& y_v \geq 0 & & & \forall\ v \in V. &
\end{aligned}
$$

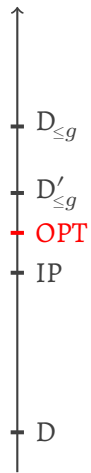
As coisas estão ficando estranhas...

- separação mais simples: encontrar árvore T de custo g e com maior peso
- é o Budget Steiner Tree Problem (BSTP) e é NP-difícil. Oh Deus!
- mas não tão difícil dessa vez, admite uma 5-aproximação!

Alinhando um novo objetivo

Um cenário possível

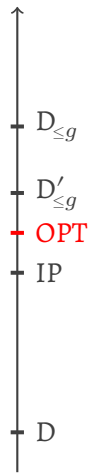
- Há diversos programas lineares envolvidos



Alinhando um novo objetivo

Um cenário possível

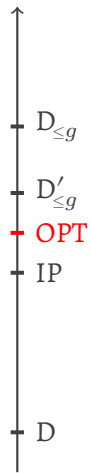
- Há diversos programas lineares envolvidos
- Não sabemos resolver nenhum deles



Alinhando um novo objetivo

Um cenário possível

- Há diversos programas lineares envolvidos
- Não sabemos resolver nenhum deles
- Mas podemos aproximar ($D'_{\leq g}$)!



Aproximando ($D'_{\leq g}$)

Executamos o método dos elipsóides parcialmente:

1. Comece com um conjunto vazio de desigualdades $\mathscr{A}$.

Aproximando ($D'_{\leq g}$)

Executamos o método dos elipsóides parcialmente:

1. Comece com um conjunto vazio de desigualdades $\mathscr{A}$.
2. Encontre uma 5-aproximação T para BSTP para a solução atual y:

Aproximando ($D'_{\leq g}$)

Executamos o método dos elipsóides parcialmente:

1. Comece com um conjunto vazio de desigualdades $\mathscr{A}$.
2. Encontre uma 5-aproximação T para BSTP para a solução atual y:
   a) Se $y(A_j) > g$, adicione a $\mathscr{A}$ a restrição correspondente a T:

$$\sum_{v \in S} y_v \leq g$$

Aproximando ($D'_{\leq g}$)

Executamos o método dos elipsóides parcialmente:

1. Comece com um conjunto vazio de desigualdades $\mathscr{A}$.
2. Encontre uma 5-aproximação T para BSTP para a solução atual y:
   a) Se $y(A_j) > g$, adicione a $\mathscr{A}$ a restrição correspondente a T:

$$\sum_{v \in S} y_v \leq g$$

   b) Se $y(A_j) \leq g$, pare e devolva a solução y e o conjunto $\mathscr{A}$.

Aproximando ($D'_{\leq g}$)

Executamos o método dos elipsóides parcialmente:

1. Comece com um conjunto vazio de desigualdades $\mathscr{A}$.
2. Encontre uma 5-aproximação T para BSTP para a solução atual y:
   a) Se $y(A_j) > g$, adicione a $\mathscr{A}$ a restrição correspondente a T:

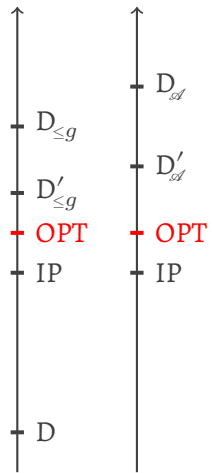
$$\sum_{v \in S} y_v \leq g$$

   b) Se $y(A_j) \leq g$, pare e devolva a solução y e o conjunto $\mathscr{A}$.
3. Resolva a relaxação original com variáveis correspondentes a $\mathscr{A}$.

Limitando o valor do PL aproximado

Voltando a figura

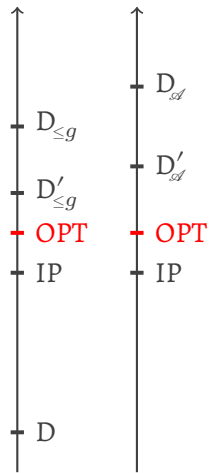
- Devemos estabelecer alguns limitantes:



Limitando o valor do PL aproximado

Voltando a figura

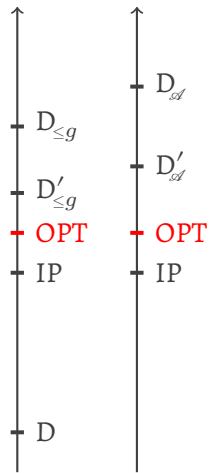
- Devemos estabelecer alguns limitantes:
  - $\mathrm{OPT}(\mathrm{D}_{\mathscr{A}}) \leq 2 \cdot \mathrm{OPT}(\mathrm{D}'_{\mathscr{A}})$



Limitando o valor do PL aproximado

Voltando a figura

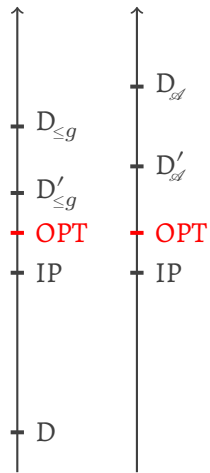
- Devemos estabelecer alguns limitantes:
  - $\mathrm{OPT}(\mathrm{D}_{\mathscr{A}}) \leq 2 \cdot \mathrm{OPT}(\mathrm{D}'_{\mathscr{A}})$
  - $\mathrm{OPT}(\mathrm{D}'_{\mathscr{A}}) \leq 5 \cdot \mathrm{OPT}(\mathrm{D}'_{\leq g})$



Limitando o valor do PL aproximado

Voltando a figura

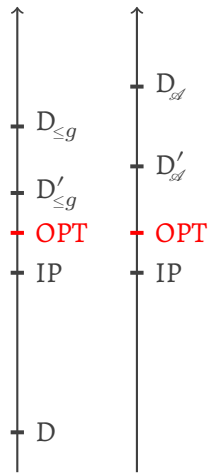
- Devemos estabelecer alguns limitantes:
  - $\mathrm{OPT}(\mathrm{D}_{\mathscr{A}}) \leq 2 \cdot \mathrm{OPT}(\mathrm{D}'_{\mathscr{A}})$
  - $\mathrm{OPT}(\mathrm{D}'_{\mathscr{A}}) \leq 5 \cdot \mathrm{OPT}(\mathrm{D}'_{\leq g})$
  - $\mathrm{OPT}(\mathrm{D}'_{\leq g}) \leq \mathrm{OPT}(\mathrm{D}_{\leq g})$



Limitando o valor do PL aproximado

Voltando a figura

- Devemos estabelecer alguns limitantes:
  - $\mathrm{OPT}(\mathrm{D}_{\mathscr{A}}) \leq 2 \cdot \mathrm{OPT}(\mathrm{D}'_{\mathscr{A}})$
  - $\mathrm{OPT}(\mathrm{D}'_{\mathscr{A}}) \leq 5 \cdot \mathrm{OPT}(\mathrm{D}'_{\leq g})$
  - $\mathrm{OPT}(\mathrm{D}'_{\leq g}) \leq \mathrm{OPT}(\mathrm{D}_{\leq g})$
  - $\mathrm{OPT}(\mathrm{P}) \leq 4 \cdot \mathrm{OPT}(\mathrm{IP})$



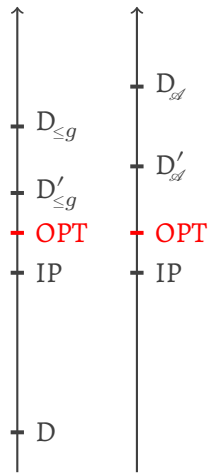
Limitando o valor do PL aproximado

Voltando a figura

- Devemos estabelecer alguns limitantes:
  - $\mathrm{OPT}(\mathrm{D}_{\mathscr{A}}) \leq 2 \cdot \mathrm{OPT}(\mathrm{D}'_{\mathscr{A}})$
  - $\mathrm{OPT}(\mathrm{D}'_{\mathscr{A}}) \leq 5 \cdot \mathrm{OPT}(\mathrm{D}'_{\leq g})$
  - $\mathrm{OPT}(\mathrm{D}'_{\leq g}) \leq \mathrm{OPT}(\mathrm{D}_{\leq g})$
  - $\mathrm{OPT}(\mathrm{P}) \leq 4 \cdot \mathrm{OPT}(\mathrm{IP})$

- Isso soma em

$$\mathrm{OPT}(\mathrm{D}_{\mathscr{A}}) \leq O(\mathrm{OPT}(\mathrm{IP}))$$



Concluindo

- Existe uma $O(\log n)$-aproximação para UCARS.
- Ela é assintoticamente ótima a não ser que $\mathrm{P} = \mathrm{NP}$.

Concluindo

- Existe uma $O(\log n)$-aproximação para UCARS.
- Ela é assintoticamente ótima a não ser que P $=$ NP.

Extensões:

- um framework para problemas com múltiplas distâncias
- generaliza diversos problemas (Prize-Collecting Steiner Tree, Constrained Forest, etc.)
- a aparecer no Latin 2020

Concluindo

- Existe uma $O(\log n)$-aproximação para UCARS.
- Ela é assintoticamente ótima a não ser que P $=$ NP.

Extensões:

- um framework para problemas com múltiplas distâncias
- generaliza diversos problemas (Prize-Collecting Steiner Tree, Constrained Forest, etc.)
- a aparecer no Latin 2020

Questão em aberto: qual é a dificuldade se o número de carros é constante?

Obrigado!

lehilton@ic.unicamp.br