

Tópicos em Otimização II

Otimização aplicada ao Roteamento de Veículos

Semestre 02/2018

Profa. Dra. Franklina Toledo (fran@icmc.usp.br)

Profa. Dra. Maristela Santos (mari@icmc.usp.br)

Prof. Dr. Pedro Munari (munari@dep.ufscar.br)

Prof. Dr. Victor Camargo (victor.camargo@dep.ufscar.br)

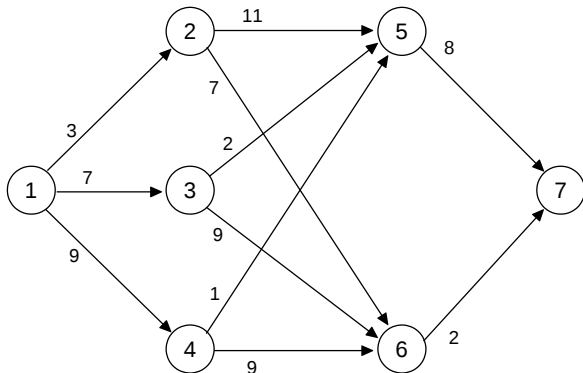
Aula 1: Introdução à disciplina; Problema do caminho mínimo; Problema do caixeiro viajante

Objetivos da aula de hoje

- ▶ Introduzir os tópicos, cronograma proposto e critérios de avaliação da disciplina;
- ▶ Estudar o problema do caminho mínimo e suas aplicações;
- ▶ Estudar o problema do caixeiro viajante e suas aplicações.

Problema do caminho mínimo (PCM)

► Lista 1, Exercício 1: Qual o caminho mínimo de 1 a 7?



Problema do caminho mínimo (PCM)

▷ Definição

Dentre várias cidades disponíveis, há uma de origem e uma de destino. Deve-se determinar qual o caminho de menor custo para ir da origem ao destino, podendo passar pelas outras cidades.

Problema do caminho mínimo (PCM)

▷ Definição

Dentre várias cidades disponíveis, há uma de origem e uma de destino. Deve-se determinar qual o caminho de menor custo para ir da origem ao destino, podendo passar pelas outras cidades.

- ▶ $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;

Problema do caminho mínimo (PCM)

▷ Definição

Dentre várias cidades disponíveis, há uma de origem e uma de destino. Deve-se determinar qual o caminho de menor custo para ir da origem ao destino, podendo passar pelas outras cidades.

- ▶ $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;
- ▶ 1 é origem, n é destino;

Problema do caminho mínimo (PCM)

► Definição

Dentre várias cidades disponíveis, há uma de origem e uma de destino. Deve-se determinar qual o caminho de menor custo para ir da origem ao destino, podendo passar pelas outras cidades.

- $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;
- 1 é origem, n é destino;
- Existe um custo de viagem c_{ij} entre cada par (i, j) , com $i, j \in \mathcal{I}$ tal que $i \prec j$ (i precede j);

Problema do caminho mínimo (PCM)

► Definição

Dentre várias cidades disponíveis, há uma de origem e uma de destino. Deve-se determinar qual o caminho de menor custo para ir da origem ao destino, podendo passar pelas outras cidades.

- $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;
- 1 é origem, n é destino;
- Existe um custo de viagem c_{ij} entre cada par (i, j) , com $i, j \in \mathcal{I}$ tal que $i \prec j$ (i precede j);
- Não há obrigação de visitar as demais cidades;

Problema do caminho mínimo (PCM)

▷ Definição

Dentre várias cidades disponíveis, há uma de origem e uma de destino. Deve-se determinar qual o caminho de menor custo para ir da origem ao destino, podendo passar pelas outras cidades.

- ▶ $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;
- ▶ 1 é origem, n é destino;
- ▶ Existe um custo de viagem c_{ij} entre cada par (i, j) , com $i, j \in \mathcal{I}$ tal que $i \prec j$ (i precede j);
- ▶ Não há obrigação de visitar as demais cidades;
- ▶ Diversas formas de resolver: programação dinâmica; algoritmo de Dijkstra; **otimização linear inteira**; heurísticas.

Problema do caminho mínimo (PCM)

▷ Modelagem

Variáveis de decisão:

Problema do caminho mínimo (PCM)

▷ Modelagem

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{caminho vai da cidade } i \text{ para a } j \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Problema do caminho mínimo (PCM)

▷ Modelagem

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{caminho vai da cidade } i \text{ para a } j \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Função objetivo:

Problema do caminho mínimo (PCM)

► Modelagem

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{caminho vai da cidade } i \text{ para a } j \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Função objetivo:

- Minimizar o custo do caminho:

Problema do caminho mínimo (PCM)

► Modelagem

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{caminho vai da cidade } i \text{ para a } j \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Função objetivo:

► Minimizar o custo do caminho: $\min \sum_{i \in \mathcal{I}} \sum_{\substack{j \in \mathcal{I} \\ i \prec j}} c_{ij} x_{ij};$

Problema do caminho mínimo (PCM)

▷ Restrições

Problema do caminho mínimo (PCM)

▷ Restrições

- ▶ O caminho deve iniciar na cidade de origem:

Problema do caminho mínimo (PCM)

▷ Restrições

- ▶ O caminho deve iniciar na cidade de origem:

$$\sum_{\substack{j \in \mathcal{I}: \\ 1 \prec j}} x_{1j} = 1$$

Problema do caminho mínimo (PCM)

▷ Restrições

- ▶ O caminho deve iniciar na cidade de origem:

$$\sum_{\substack{j \in \mathcal{I}: \\ 1 \prec j}} x_{1j} = 1$$

- ▶ O caminho deve terminar na cidade de destino:

Problema do caminho mínimo (PCM)

▷ Restrições

- ▶ O caminho deve iniciar na cidade de origem:

$$\sum_{\substack{j \in \mathcal{I}: \\ 1 \prec j}} x_{1j} = 1$$

- ▶ O caminho deve terminar na cidade de destino:

$$\sum_{\substack{i \in \mathcal{I}: \\ i \prec n}} x_{in} = 1$$

- ▶ Restrição de **fluxo** (garante entrada e saída da cidade):

Problema do caminho mínimo (PCM)

▷ Restrições

- ▶ O caminho deve iniciar na cidade de origem:

$$\sum_{\substack{j \in \mathcal{I}: \\ 1 \prec j}} x_{1j} = 1$$

- ▶ O caminho deve terminar na cidade de destino:

$$\sum_{\substack{i \in \mathcal{I}: \\ i \prec n}} x_{in} = 1$$

- ▶ Restrição de **fluxo** (garante entrada e saída da cidade):

$$\sum_{\substack{i \in \mathcal{I}: \\ i \prec h}} x_{ih} = \sum_{\substack{j \in \mathcal{I}: \\ h \prec j}} x_{hj}, \quad \forall h \in \mathcal{I} \setminus \{1, n\}$$

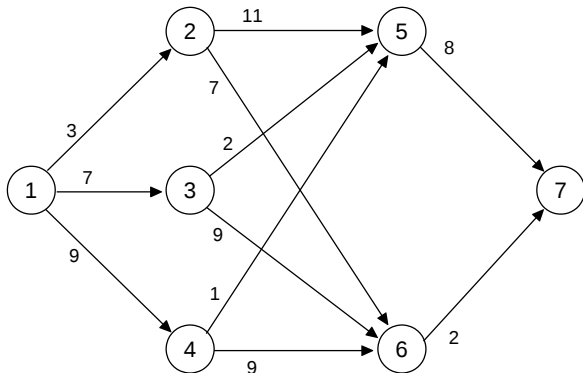
Problema do caminho mínimo (PCM)

▷ Modelo algébrico

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{I}} \sum_{\substack{j \in \mathcal{I} \\ i \prec j}} c_{ij} x_{ij} \\ \text{s.a} \quad & \sum_{\substack{j \in \mathcal{I}: \\ 1 \prec j}} x_{1j} = 1, \\ & \sum_{\substack{i \in \mathcal{I}: \\ i \prec n}} x_{in} = 1, \\ & \sum_{\substack{i \in \mathcal{I}: \\ i \prec h}} x_{ih} = \sum_{\substack{j \in \mathcal{I}: \\ h \prec j}} x_{hj}, \quad \forall h \in \mathcal{I} \setminus \{1, n\}, \\ & x_{ij} \in \{0, 1\}, \quad i, j \in \mathcal{I}. \end{aligned}$$

Problema do caminho mínimo (PCM)

► Lista 1, Exercício 1: Qual o caminho mínimo de 1 a 7?



Problema do caixeiro viajante (PCV)

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Observações:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Observações:

- ▶ Nome original: *Traveling salesman problem* (TSP);

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Observações:

- ▶ Nome original: *Traveling salesman problem* (TSP);
- ▶ “Caixeiro-viajante é uma profissão **antiga**, de uma pessoa que vende produtos fora de onde eles são produzidos.

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Observações:

- ▶ Nome original: *Traveling salesman problem* (TSP);
- ▶ “Caixeiro-viajante é uma profissão **antiga**, de uma pessoa que vende produtos fora de onde eles são produzidos. **Antigamente**, quando não havia a facilidade do transporte entre cidades, os caixeiros-viajantes eram a única forma de transportar produtos entre diferentes regiões fora das grandes cidades.”

Problema do caixeiro viajante (PCV)

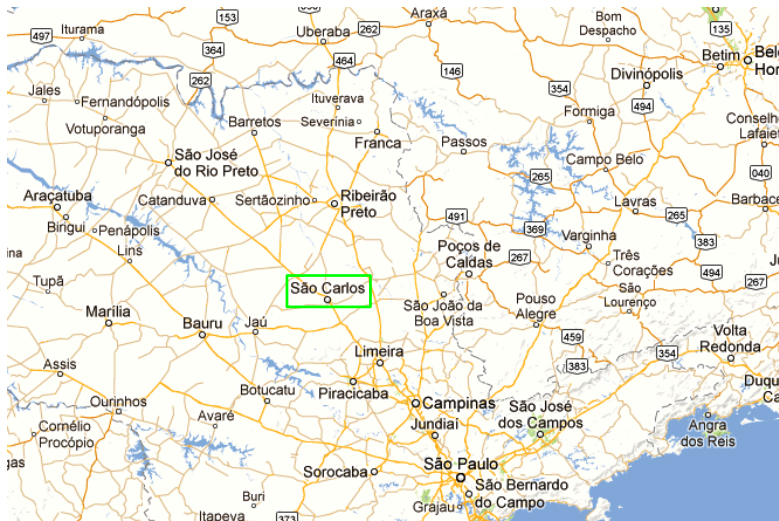
Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Observações:

- ▶ Nome original: *Traveling salesman problem* (TSP);
- ▶ “Caixeiro-viajante é uma profissão **antiga**, de uma pessoa que vende produtos fora de onde eles são produzidos. **Antigamente**, quando não havia a facilidade do transporte entre cidades, os caixeiros-viajantes eram a única forma de transportar produtos entre diferentes regiões fora das grandes cidades.”
- ▶ Atualmente, este problema modela diversas situações reais.

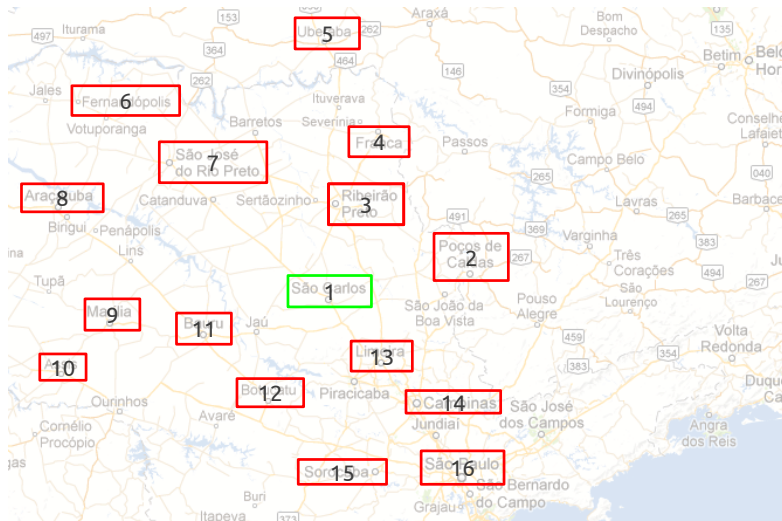
Problema do caixeiro viajante (PCV)



Problema do caixeiro viajante (PCV)



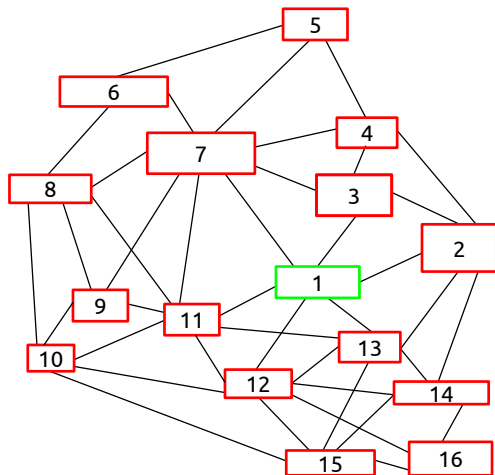
Problema do caixeiro viajante (PCV)



A map of Minas Gerais, Brazil, with 16 numbered nodes (1-16) representing cities. The nodes are connected by lines representing roads. Node 1 (São Carlos) is highlighted with a green border, while all other nodes (2-16) have red borders. The connections between nodes are as follows:

- Node 1 is connected to nodes 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 2 is connected to nodes 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 3 is connected to nodes 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 4 is connected to nodes 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 5 is connected to nodes 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 6 is connected to nodes 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 7 is connected to nodes 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 8 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, and 16.
- Node 9 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, and 16.
- Node 10 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, and 16.
- Node 11 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, and 16.
- Node 12 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, and 16.
- Node 13 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, and 16.
- Node 14 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, and 16.
- Node 15 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 16.
- Node 16 is connected to nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16.

Problema do caixeiro viajante (PCV)



Grafo $G = (V, A)$

V: vértices ou nós

A: arestas

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Parâmetros:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Parâmetros:

- ▶ $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Parâmetros:

- ▶ $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;
- ▶ D_{ij} : distância entre as cidades i e j , para todo $i, j \in \mathcal{I}$.

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Parâmetros:

- ▶ $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;
- ▶ D_{ij} : distância entre as cidades i e j , para todo $i, j \in \mathcal{I}$.

Objetivo:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Parâmetros:

- ▶ $\mathcal{I} = \{1, 2, \dots, n\}$: cidades;
- ▶ D_{ij} : distância entre as cidades i e j , para todo $i, j \in \mathcal{I}$.

Objetivo:

- ▶ Determinar uma rota que passe por todas as cidades e volte para a cidade inicial, de modo a minimizar a distância percorrida.

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Variáveis de decisão:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{vendedor vai da cidade } i \in \mathcal{I} \text{ para a } j \in \mathcal{I} \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{vendedor vai da cidade } i \in \mathcal{I} \text{ para a } j \in \mathcal{I} \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Função objetivo:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{vendedor vai da cidade } i \in \mathcal{I} \text{ para a } j \in \mathcal{I} \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Função objetivo:

- ▶ Minimizar a distância:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{vendedor vai da cidade } i \in \mathcal{I} \text{ para a } j \in \mathcal{I} \text{ imediatamente,} \\ 0, & \text{caso contrário.} \end{cases}$$

Função objetivo:

- ▶ Minimizar a distância: $\min \sum_{i \in \mathcal{I}} \sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} D_{ij} x_{ij};$

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Cada cidade deve ser visitada uma única vez:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Cada cidade deve ser visitada uma única vez:

$$\sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} x_{ij} = 1, \quad i \in \mathcal{I}$$

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Cada cidade deve ser visitada uma única vez:

$$\sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} x_{ij} = 1, \quad i \in \mathcal{I}$$

- ▶ Isso garante que cada cidade tem uma única sucessora imediata.

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Cada cidade deve ser visitada uma única vez:

$$\sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} x_{ij} = 1, \quad i \in \mathcal{I}$$

- ▶ Isso garante que cada cidade tem uma única sucessora imediata.
Também garante única antecessora?

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Cada cidade tem uma única antecessora imediata:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

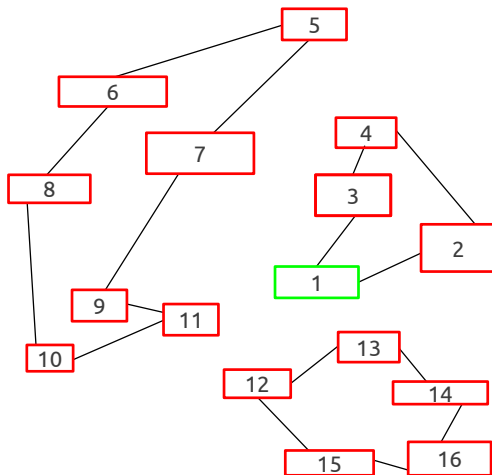
- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Cada cidade tem uma única antecessora imediata:

$$\sum_{\substack{i \in \mathcal{I}: \\ i \neq j}} x_{ij} = 1, \quad j \in \mathcal{I}$$

Problema do caixeiro viajante (PCV)



Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Evita sub-rota:

Problema do caixeiro viajante (PCV)

Um vendedor precisa visitar seus clientes em n cidades.

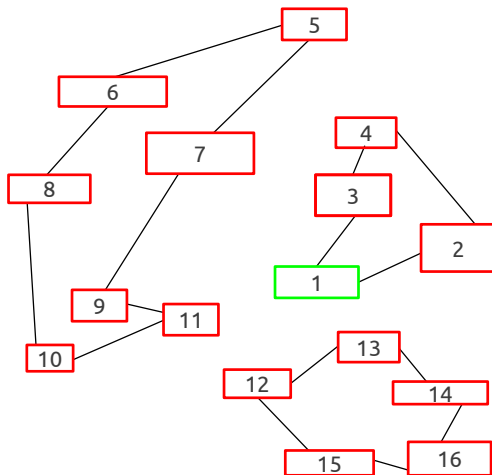
- ▶ Ele parte de uma cidade e deve retornar a essa cidade após visitar todas as outras.
- ▶ Cada cidade deve ser visitada uma única vez.

Restrições:

- ▶ Evita sub-rota:

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij} \leq |S| - 1, \quad \forall S \subset \mathcal{I}, \quad 1 < |S| < n$$

Problema do caixeiro viajante (PCV)



Problema do caixeiro viajante (PCV)

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{I}} \sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} D_{ij} x_{ij} \\ \text{s.a} \quad & \sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} x_{ij} = 1, \quad i \in \mathcal{I}, \\ & \sum_{\substack{i \in \mathcal{I}: \\ i \neq j}} x_{ij} = 1, \quad j \in \mathcal{I}, \\ & \sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij} \leq |S| - 1, \quad \forall S \subset \mathcal{I}, \quad 1 < |S| < n, \\ & x_{ij} \in \{0, 1\}, \quad i, j \in \mathcal{I}. \end{aligned}$$

Problema do caixeiro viajante (PCV)

- Formulação de Dantzig, Fulkerson e Johnson (DFJ);

Problema do caixeiro viajante (PCV)

- ▶ Formulação de Dantzig, Fulkerson e Johnson (DFJ);
- ▶ Possui como desvantagem o número de restrições: da ordem de 2^n ;

Problema do caixeiro viajante (PCV)

- ▶ Formulação de Dantzig, Fulkerson e Johnson (DFJ);
- ▶ Possui como desvantagem o número de restrições: da ordem de 2^n ;
- ▶ Para 16 cidades: mais de 65536 restrições; Para 100 cidades: mais de 10^{30} restrições;

Problema do caixeiro viajante (PCV)

- ▶ Formulação de Dantzig, Fulkerson e Johnson (DFJ);
- ▶ Possui como desvantagem o número de restrições: da ordem de 2^n ;
- ▶ Para 16 cidades: mais de 65536 restrições; Para 100 cidades: mais de 10^{30} restrições;
- ▶ Pode se restringir os conjuntos a:

$$2 \leq |S| \leq \left\lfloor \frac{n}{2} \right\rfloor$$

mas ainda assim é exponencial :(

Problema do caixeiro viajante (PCV)

- ▶ Formulação de Dantzig, Fulkerson e Johnson (DFJ);
- ▶ Possui como desvantagem o número de restrições: da ordem de 2^n ;
- ▶ Para 16 cidades: mais de 65536 restrições; Para 100 cidades: mais de 10^{30} restrições;
- ▶ Pode se restringir os conjuntos a:

$$2 \leq |S| \leq \left\lfloor \frac{n}{2} \right\rfloor$$

mas ainda assim é exponencial :(

- ▶ Em geral, apenas um número relativamente pequeno delas é necessário :)

Problema do caixeiro viajante (PCV)

- Para resolver este modelo, inicialmente descartamos as restrições exponenciais (relaxação combinatória);

Problema do caixeiro viajante (PCV)

- ▶ Para resolver este modelo, inicialmente descartamos as restrições exponenciais (relaxação combinatória);
- ▶ Caso a solução ótima desta relaxação tenha subrotas, inserimos restrições do tipo

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij} \leq |S| - 1$$

somente para as subrotas desta solução;

Problema do caixeiro viajante (PCV)

- ▶ Para resolver este modelo, inicialmente descartamos as restrições exponenciais (relaxação combinatória);
- ▶ Caso a solução ótima desta relaxação tenha subrotas, inserimos restrições do tipo

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij} \leq |S| - 1$$

somente para as subrotas desta solução;

- ▶ Resolvemos a relaxação novamente e caso tenha novas subrotas, repetimos o processo.

Problema do caixeiro viajante (PCV)

▷ Lista 1, Exercício 2

Um vendedor de bebidas reside em São Carlos. Ele precisa visitar as seguintes cidades nos próximos dias, para apresentar a seus clientes uma nova linha de refrigerantes de baixa caloria: Franca, Ribeirão Preto, São José do Rio Preto, Poços de Caldas e Limeira. As distâncias entre as cidades são as seguintes:

De/Para	Distâncias (km)					
	SC	FR	RP	SJRP	PC	LI
SC	—	186	105	208	177	94
FR	187	—	89.8	223	255	254
RP	99.9	89	—	203	204	172
SJRP	206	220	203	—	377	295
PC	168	251	201	376	—	156
LI	86.8	255	173	293	159	—

Deseja-se determinar uma rota que visite todas as cidades e volte à cidade atual de modo que a distância total percorrida pelo vendedor seja a menor possível.

Problema do caixeiro viajante



Problema do caixeiro viajante

▷ Resolução do Exemplo 1

Resolvendo sem as restrições de sub-rota:

	SC	FR	RP	SJRP	PC	LI
SC				1.000		
FR			1.000			
RP		1.000				
SJRP	1.000					
PC						1.000
LI					1.000	

Valor ótimo = 907.800

Problema do caixeiro viajante

► Resolução do Exemplo 1

Para eliminar a sub-rota SC-SJRP-SC, adicionamos:

$$x(SC, SJRP) + x(SJRP, SC) \leq 1$$

Solução obtida:

	SC	FR	RP	SJRP	PC	LI
SC					1.000	
FR			1.000			
RP				1.000		
SJRP		1.000				
PC						1.000
LI	1.000					

Valor ótimo = 932.600

Problema do caixeiro viajante

► Resolução do Exemplo 1

Para eliminar a sub-rota SC-PC-LI-SC, adicionamos:

$$x(SC, PC) + x(SC, LI) + x(LI, PC) + x(LI, SC) + x(PC, SC) + x(PC, LI) \leq 2$$

Solução obtida:

	SC	FR	RP	SJRP	PC	LI
SC				1.000		
FR			1.000			
RP	1.000					
SJRP		1.000				
PC						1.000
LI					1.000	

Valor ótimo = 932.700

Problema do caixeiro viajante

► Resolução do Exemplo 1

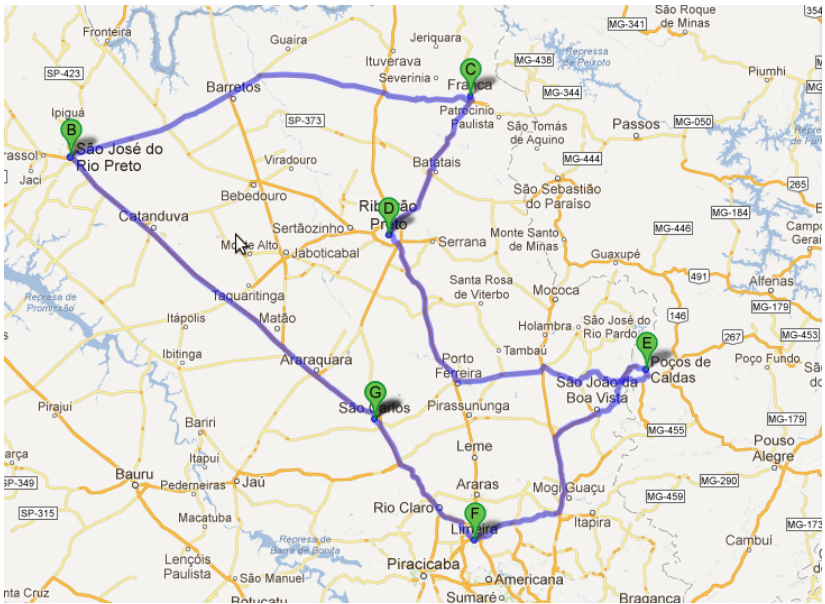
Para eliminar a sub-rota PC-LI-PC, adicionamos:

$$x(PC, LI) + x(LI, PC) \leq 1$$

Solução obtida:

	SC	FR	RP	SJRP	PC	LI
SC				1.000		
FR			1.000			
RP					1.000	
SJRP		1.000				
PC						1.000
LI	1.000					

Valor ótimo = 964.600

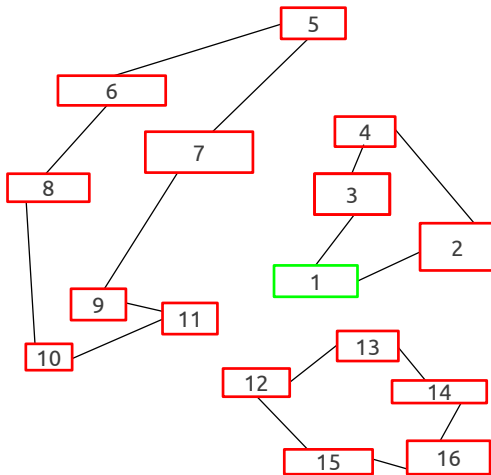


Problema do caixeiro viajante

- ▷ Formulação alternativa: número polinomial de restrições (n^2)

Problema do caixeiro viajante

▷ Formulação alternativa: número polinomial de restrições (n^2)



Problema do caixeiro viajante

▷ Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Problema do caixeiro viajante

▷ Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- ▶ Para evitar sub-rotas, vamos usar uma nova variável de decisão:

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?
 - Se i precede j imediatamente:

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?
 - Se i precede j imediatamente: $u_j \geq u_i + 1$;

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?
 - Se i precede j imediatamente: $u_j \geq u_i + 1$;
 - Caso contrário, esta restrição deve ser desligada!

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?
 - Se i precede j imediatamente: $u_j \geq u_i + 1$;
 - Caso contrário, esta restrição deve ser desligada!

- Assim, obtemos:

$$u_j \geq u_i + 1$$

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?
 - Se i precede j imediatamente: $u_j \geq u_i + 1$;
 - Caso contrário, esta restrição deve ser desligada!

- Assim, obtemos:

$$u_j \geq u_i + 1x_{ij}$$

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?
 - Se i precede j imediatamente: $u_j \geq u_i + 1$;
 - Caso contrário, esta restrição deve ser desligada!
- Assim, obtemos:

$$u_j \geq u_i + 1x_{ij} - n(1 - x_{ij}),$$

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?
 - Se i precede j imediatamente: $u_j \geq u_i + 1$;
 - Caso contrário, esta restrição deve ser desligada!
- Assim, obtemos:

$$u_j \geq u_i + 1x_{ij} - n(1 - x_{ij}), \quad i = 1, \dots, n,$$

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?

- Se i precede j imediatamente: $u_j \geq u_i + 1$;
- Caso contrário, esta restrição deve ser desligada!

- Assim, obtemos:

$$u_j \geq u_i + 1x_{ij} - n(1 - x_{ij}), \quad i = 1, \dots, n, \quad j = 2, \dots, n,$$

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?

- Se i precede j imediatamente: $u_j \geq u_i + 1$;
- Caso contrário, esta restrição deve ser desligada!

- Assim, obtemos:

$$u_j \geq u_i + 1x_{ij} - n(1 - x_{ij}), \quad i = 1, \dots, n, \quad j = 2, \dots, n, \quad i \neq j,$$

Problema do caixeiro viajante

► Formulação alternativa: Miller, Tucker e Zemlin (MTZ)

Restrições:

- Para evitar sub-rotas, vamos usar uma nova variável de decisão:

u_i : número de visitas realizadas até o nó $i \in \mathcal{I}$.

- Assim, dados dois nós i e j , o que devemos garantir?

- Se i precede j imediatamente: $u_j \geq u_i + 1$;
- Caso contrário, esta restrição deve ser desligada!

- Assim, obtemos:

$$u_j \geq u_i + 1x_{ij} - n(1 - x_{ij}), \quad i = 1, \dots, n, \quad j = 2, \dots, n, \quad i \neq j,$$

$$u_1 = 0.$$

Problema do caixeiro viajante

► Formulação alternativa

$$\min \quad \sum_{i \in \mathcal{I}} \sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} D_{ij} x_{ij}$$

$$\text{s.a} \quad \sum_{\substack{j \in \mathcal{I}: \\ j \neq i}} x_{ij} = 1, \quad i \in \mathcal{I},$$

$$\sum_{\substack{i \in \mathcal{I}: \\ i \neq j}} x_{ij} = 1, \quad j \in \mathcal{I},$$

$$u_j \geq u_i + x_{ij} - n(1 - x_{ij}), \quad i \in \mathcal{I}, j \in \mathcal{I} \setminus \{1\}, i \neq j$$

$$u_1 = 0,$$

$$x_{ij} \in \{0, 1\}, \quad i, j \in \mathcal{I},$$

$$u_i \geq 0, \quad i \in \mathcal{I}.$$

Problema do caixeiro viajante

▷ Lista 1, Exercício 2

Um vendedor de bebidas reside em São Carlos. Ele precisa visitar as seguintes cidades nos próximos dias, para apresentar a seus clientes uma nova linha de refrigerantes de baixa caloria: Franca, Ribeirão Preto, São José do Rio Preto, Poços de Caldas e Limeira. As distâncias entre as cidades são as seguintes:

De/Para	Distâncias (km)					
	SC	FR	RP	SJRP	PC	LI
SC	—	186	105	208	177	94
FR	187	—	89.8	223	255	254
RP	99.9	89	—	203	204	172
SJRP	206	220	203	—	377	295
PC	168	251	201	376	—	156
LI	86.8	255	173	293	159	—

Deseja-se determinar uma rota que visite todas as cidades e volte à cidade atual de modo que a distância total percorrida pelo vendedor seja a menor possível.

Problema do caixeiro viajante

▷ Lista 1, Exercício 2

	SC	FR	RP	SJRP	PC	LI
SC				1.000		
FR			1.000			
RP					1.000	
SJRP		1.000				
PC						1.000
LI	1.000					

Valor ótimo = 964.600

Problema do caixeiro viajante (PCV)

▷ [Links interessantes](#)

Problema do caixeiro viajante (PCV)

▷ Links interessantes

- ▶ PCV no Google Maps:

<http://www.tsp.gatech.edu/maps/index.html>

- ▶ História, aplicações e curiosidades do PCV:

<http://www.tsp.gatech.edu/>

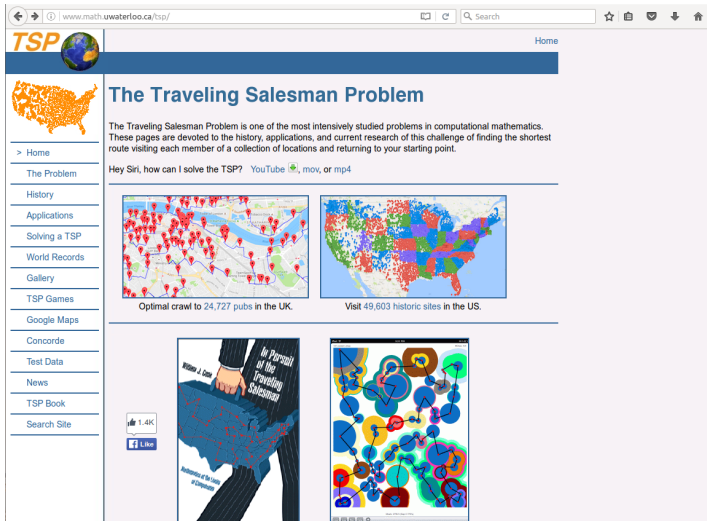
http://en.wikipedia.org/wiki/Travelling_salesman_problem

- ▶ Ilustração de heurística com animação gráfica:

<http://www-e.uni-magdeburg.de/mertens/TSP/>

Links interessantes

▷ <http://www.tsp.gatech.edu>




The screenshot shows the homepage of the Traveling Salesman Problem (TSP) website. The browser address bar displays www.math.uwaterloo.ca/tsp/. The website features a blue header with the 'TSP' logo and a globe. A sidebar on the left contains a navigation menu with links: Home, The Problem, History, Applications, Solving a TSP, World Records, Gallery, TSP Games, Google Maps, Concorde, Test Data, News, TSP Book, and Search Site. The main content area is titled 'The Traveling Salesman Problem' and includes a paragraph explaining the problem. Below this, there are two images: a map of the UK with red pins and a map of the US with colored regions. At the bottom, there are two more images: a book cover titled 'In Pursuit of the Traveling Salesman' and a colorful graph visualization. Social media links for YouTube, Google+, and mp4 are also present.

Home

The Traveling Salesman Problem

The Traveling Salesman Problem is one of the most intensively studied problems in computational mathematics. These pages are devoted to the history, applications, and current research of this challenge of finding the shortest route visiting each member of a collection of locations and returning to your starting point.

Hey Siri, how can I solve the TSP? YouTube , mov, or mp4

Optimal crawl to 24,727 pubs in the UK.

Visit 49,603 historic sites in the US.

In Pursuit of the Traveling Salesman

Mathematics of the Traveling Salesman Problem

1.4K Likes

Links interessantes

▷ <http://www-e.uni-magdeburg.de/mertens/TSP/>

The screenshot shows a web browser window with the URL `www-e.uni-magdeburg.de/mertens/TSP/node2.html`. The page title is "2. Construction Heuristics". The content describes construction heuristics and lists three types: Nearest Neighbor Heuristic (2.1), Insertion Heuristics (2.2), and Savings Heuristic (in the near future). Below this, section "2.1 Nearest Neighbor Heuristic" explains the algorithm and mentions a demo. The demo interface includes buttons for "new", "reset", "step", and "stop", a "Nodes:" section with radio buttons for 10, 15, 20, and 25, and a "Speed" section with radio buttons for "fast", "med.", and "slow". A "solve" button is at the bottom left. The main area shows a graph with 10 nodes and a tour. Below the graph, it states: "Nearest neighbor tour has length 1.1159" and "Optimum tour has length 0.898".

2. Construction Heuristics

A construction heuristic is an algorithm that determines a tour according to some construction rules, but does not try to improve upon this tour. A tour is successively built and parts already built remain unchanged throughout the algorithm. A detailed discussion of construction heuristics can be found in [2].

We will discuss

- Nearest Neighbor Heuristic [2.1](#)
- Insertion Heuristics [2.2](#)
- Savings Heuristic (in the near future)

2.1 Nearest Neighbor Heuristic

The salesman starts at some city and then visits the city nearest to the starting city. From there he visits the nearest city that was not visited so far, etc., until all cities are visited, and the salesman returns to the start.

This is the first heuristic that almost everyone comes up with. It is probably close the real salesman's approach. It is a poor heuristic, however. As can be seen by playing with the demo below, several cities are "forgotten" during the course of the algorithm. They have to be inserted at high costs in the end.

TSP: Nearest Neighbour Heuristic
(C) 1999, Stephan.Mertens@Physik.Uni-Magdeburg.DE

new
reset
step
stop

Nodes:
☒ 10
☐ 15
☐ 20
☐ 25

Speed
☐ fast
☒ med.
☐ slow

solve

Nearest neighbor tour has length 1.1159
Optimum tour has length 0.898

Outras formulações

▷ Oncan et al. 2009



Available online at www.sciencedirect.com



ScienceDirect

Computers & Operations Research 36 (2009) 637–654

computers &
operations
research

www.elsevier.com/locate/cor

Invited review

A comparative analysis of several asymmetric traveling salesman problem formulations

Temel Öncan^a, İ. Kuban Altınel^b, Gilbert Laporte^{c,*}

^aDepartment of Industrial Engineering, Galatasaray University, Ortaköy, İstanbul 34357, Turkey

^bDepartment of Industrial Engineering, Boğaziçi University, Bebek, İstanbul 34342, Turkey

^cCanada Research Chair in Distribution Management, HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7

Available online 3 December 2007

Abstract

In this survey, a classification of 24 asymmetric traveling salesman problem (ATSP) formulations is presented. The strength of their LP relaxations is discussed and known relationships from the literature are reviewed. Some new relationships are also introduced, and computational results are reported.

© 2007 Elsevier Ltd. All rights reserved.

Keywords: Integer linear programming; Asymmetric traveling salesman problem; Formulations; Projections

- ▶ Obrigado pela atenção!
- ▶ Dúvidas?
- ▶ Próxima aula:
 - ▶ Problema do caixeiro viajante (outras formulações);
 - ▶ Problema de roteamento de veículos;