Simulação e Otimização

Capítulo 2: Problemas de otimização combinatória

Raquel Bernardino

rbernardino@iseg.ulisboa.pt
Gabinete 511 Quelhas 6

Sumário

Sumário

Motivação

Dado:

- um conjunto de cidades N; e
- custos de deslocação entre cada par de cidades c_{ij}, $i, j \in N : i \neq j$,

o Problema do Caixeiro Viajante consiste em determinar um circuito (ou ciclo) de menor custo que passe por todas as cidades uma e uma só vez.

Motivação

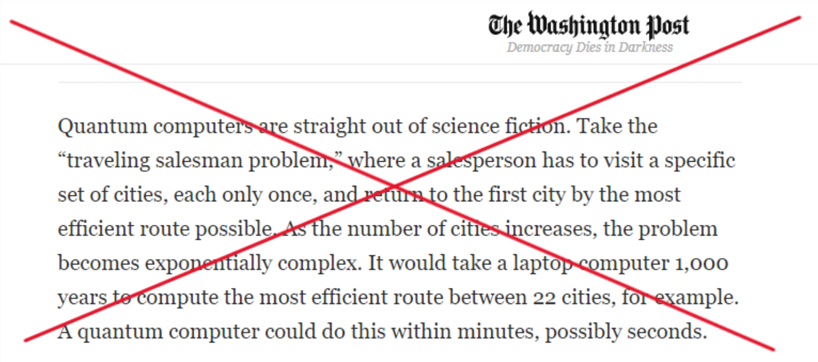
The Washington Post
Democracy Dies in Darkness

Quantum computers are straight out of science fiction. Take the "traveling salesman problem," where a salesperson has to visit a specific set of cities, each only once, and return to the first city by the most efficient route possible. As the number of cities increases, the problem becomes exponentially complex. It would take a laptop computer 1,000 years to compute the most efficient route between 22 cities, for example. A quantum computer could do this within minutes, possibly seconds.

Figura 1: Notícia errada sobre o problema caixeiro viajante no "The Washington Post".

Palestra de William Cook sobre o problema do caixeiro viajante:
`https://www.youtube.com/watch?v=5VjphFYQKj8&t=55s` [Euro 2019, Dublin]

Site com um *solver* para o caixeiro viajante:
`https://www.math.uwaterloo.ca/tsp/index.html`


The Washington Post
Democracy Dies in Darkness

~~Quantum computers are straight out of science fiction. Take the "traveling salesman problem," where a salesperson has to visit a specific set of cities, each only once, and return to the first city by the most efficient route possible. As the number of cities increases, the problem becomes exponentially complex. It would take a laptop computer 1,000 years to compute the most efficient route between 22 cities, for example. A quantum computer could do this within minutes, possibly seconds.~~

Figura 2: Notícia errada sobre o problema caixeiro viajante no "The Washington Post".

Palestra de William Cook sobre o problema do caixeiro viajante: `https://www.youtube.com/watch?v=5VjphFYQKj8&t=55s` [Euro 2019, Dublin]

Site com um *solver* para o caixeiro viajante: `https://www.math.uwaterloo.ca/tsp/index.html`

- 1954 Dantzig, Ford & Fulkerson [1] 49 cidades

Figura 3: O problema do caixeiro viajante com 49 cidades.

[1] Dantzig, G., Fulkerson, R., & Johnson, S. (1954). *Solution of a large-scale traveling-salesman problem*. Journal of the Operations Research Society of America, 2(4), 393-410.

Sumário

Introdução

- Os problemas de roteamento consistem em estabelecer rotas com determinadas características de forma a otimizar um determinado critério.

- Metodologia:
  - Métodos exatos
    - Modelos de programação linear inteira (mista). → Foco do capítulo.
  - Métodos aproximados
    - Relaxações.
    - Heurísticas.

- Os métodos exatos serão implementados utilizando software específico.
  - O pacote PuLP do Python com o *solver* da COIN-OR resolve os modelos de PLIM.
    - Exemplos de outros *solvers* são: CPLEX, Gurobi, Xpress, OpenSolver.

Introdução

- Definição genérica de um problema de roteamento

Queremos determinar uma rota que:

- comece e termine num ponto,
- passe por um subconjunto de clientes satisfazendo as suas necessidades, e
- minimize o "custo".

- Uma rota é uma sequência de vértices que começa e acaba num depósito.

Características dos problemas de roteamento

- Rotas:
  - Rotas fechadas ou abertas (começam e acabam em depósitos diferentes).
  - Rotas equilibradas - visitam o mesmo número de clientes.
  - Rotas “giras” - compactas, não se intersetam.

- Clientes:
  - Apenas alguns clientes requerem serviço.
  - Clientes têm que ser servidos num determinado período de tempo - janela temporal.
  - Um tipo de clientes pode ter prioridade sobre outros tipos de clientes - restrições de precedência.
  - Clientes têm oferta que é usada para satisfazer a procura de outros clientes - *pick-up and delivery*.
  - Clientes requerem visitas periódicas ou consistentes.

Características dos problemas de roteamento

- Frota:
  - Um ou vários veículos.
    - Com vários veículos: exatamente, pelo menos ou no máximo.
  - É homogénea ou heterogénea.
  - Veículos têm limites à capacidade do que podem transportar ou à distância percorrida - veículos elétricos.
  - Cada veículo pode fazer uma ou várias rotas.

- Objetivos:
  - Minimizar o custo.
  - Minimizar o número de veículos utilizados.
  - Minimizar a emissão de gases.
  - Maximizar o lucro recolhido.

Aplicações práticas

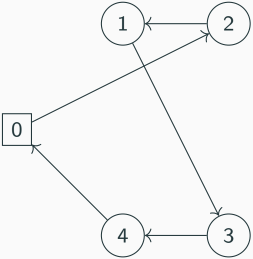
- Exemplos de aplicações práticas de problemas de roteamento são:
  - Problemas de distribuição,
  - Problemas de recolha,
  - Sequenciamento de ADN,
  - Escalonamento de cirurgias por médico,
  - Transporte de cavalos da GNR,
  - Operações num porto marítimo - escalonamento da grua,
  - Cuidados de saúde em casa,
  - Roteamento de drones,
  - Recolha de lixo,
  - Remoção de neve.
- Os últimos problemas são modelados como problemas de roteamento nos arcos.

Problemas de roteamento

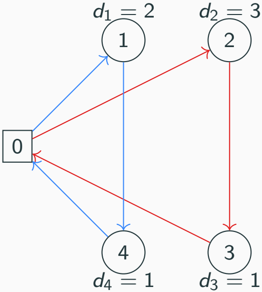
- Existem inúmeras variantes de problemas de roteamento. $\Longrightarrow$ Foco nas variantes base.

- Problema do caixeiro viajante (*traveling salesman problem - TSP*)
  - uma rota;
  - visita todos os clientes uma e uma só vez; e
  - o objetivo é minimizar o custo total de deslocação.

- Problema do roteamento de veículos (*vehicle routing problem - VRP*)
  - cada um dos K veículos homogéneos disponíveis executa uma rota;
  - cada veículo tem capacidade Q;
  - cada cliente tem procura que deve ser toda satisfeita; e
  - o objetivo é minimizar o custo total de deslocação.



(a) TSP.



(b) VRP ($K = 2$ e $Q = 4$).

Figura 4: Soluções do TSP e do VRP.

Sumário

O problema do caixeiro viajante

- O problema do caixeiro viajante pode ser modelado num grafo orientado $G = (V, A)$, onde:
  - V é o conjunto das cidades que devem ser visitadas uma e uma só vez;
    - Uma das cidades $0 \in V$ é o depósito fictício - indica o início e o fim do percurso.
  - A é o conjunto dos arcos, que representam as ligações entre as cidades.
    - $A = \{(i, j) : i, j \in V \text{ e } i \neq j\}$
  - c_{ij} é o custo de atravessar o arco $(i, j) \in A$.

- Queremos determinar o circuito de custo total mínimo que passa por todos os nodos de V uma e uma só vez (circuito Hamiltoniano).

Formulação genérica

$$x_{ij} = \begin{cases} 1, & \text{se o arco } (i,j) \in A \text{ é atravessado pelo caixeiro viajante} \\ 0, & \text{caso contrário} \end{cases}$$

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} & & \\ \text{sujeito a:} \quad & \sum_{i \in V} x_{ji} = 1 & & \forall j \in V \quad \text{(um arco sai de j)} \\ & \sum_{i \in V} x_{ij} = 1 & & \forall j \in V \quad \text{(um arco entra j)} \\ & \text{Não contém subcircuitos} & & \\ & x_{ij} \in \{0,1\} & & \forall (i,j) \in A \end{aligned}$$

Nota: Subcircuitos (ou subrotas) são circuitos que não contêm o depósito.

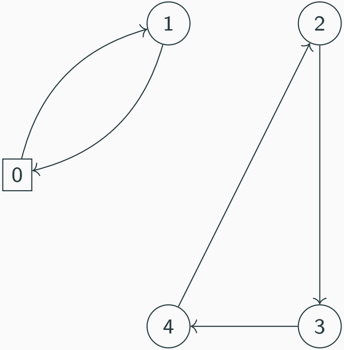


Figura 5: Solução não admissível para o TSP.

Como garantir que não temos subcircuitos?

Formulação MTZ

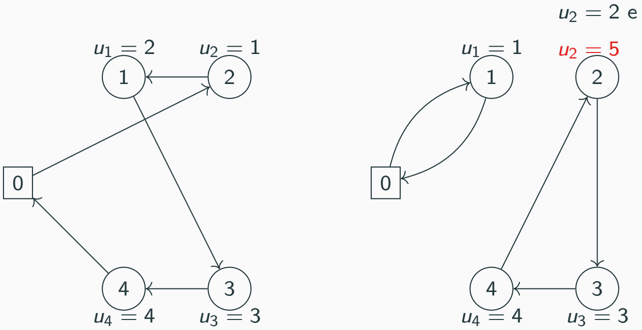
- Formulação de Miller-Tucker-Zemlin (MTZ) [2]

u_i - posição que o nodo $i \in V \setminus \{0\}$ ocupa na rota.

$$
\begin{aligned}
& \min \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{sujeito a: } & \sum_{i\in V} x_{ji} = 1 & \forall j \in V & \quad \text{(um arco sai de j)} \\
& \sum_{i\in V} x_{ij} = 1 & \forall j \in V & \quad \text{(um arco entra de j)} \\
& u_i + (|V|-1)x_{ij} \leq u_j + |V| - 2 & \forall (i,j) \in A : i,j \neq 0 & \quad \text{(posição j é a do i mais 1,} \\
& & & \quad \text{se vamos de i para j)} \\
& 1 \leq u_i \leq |V| - 1 & \forall i \in V \setminus \{0\} & \quad \text{(limites variáveis u)} \\
& x_{ij} \in \{0,1\} & \forall (i,j) \in A &
\end{aligned}
$$

[2] Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). *Integer programming formulation of traveling salesman problems*. Journal of the ACM (JACM), 7(4), 326-329.

Formulação MTZ



(a) Solução admissível TSP.

(b) Solução não admissível TSP.

Figura 6: Como funciona o MTZ.

Formulação MTZ

- Consideremos $V = \{0, 1, 2\}$.

$$
\begin{aligned}
\text{min } & c_{01}x_{01} + c_{02}x_{02} + c_{10}x_{10} + c_{12}x_{12} + c_{20}x_{20} + c_{21}x_{21} \\
\text{sujeito a: } & x_{01} + x_{02} = 1 \qquad \text{(sai 0)} \\
& x_{10} + x_{12} = 1 \qquad \text{(sai 1)} \\
& x_{20} + x_{21} = 1 \qquad \text{(sai 2)} \\
& x_{10} + x_{20} = 1 \qquad \text{(entra 0)} \\
& x_{01} + x_{21} = 1 \qquad \text{(entra 1)} \\
& x_{02} + x_{12} = 1 \qquad \text{(entra 2)} \\
& u_1 + 2x_{12} \leq u_2 + 1 \qquad \text{(posição do 2 é a do 1 mais 1, se vamos de 1 para 2)} \\
& u_2 + 2x_{21} \leq u_1 + 1 \qquad \text{(posição do 1 é a do 2 mais 1, se vamos de 2 para 1)} \\
& 1 \leq u_1, u_2 \leq 2 \qquad \text{(limites u)} \\
& x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A
\end{aligned}
$$

Formulação SCF

- Formulação de fluxo único - Single Commodity Flow (SCF) [3]

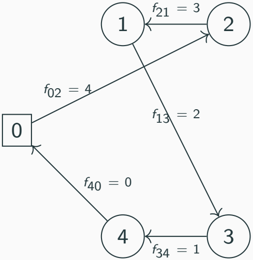
f_{ij} - fluxo que atravessa o arco $(i, j) \in A$, que corresponde ao número de nodos que ainda têm que ser visitados.

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

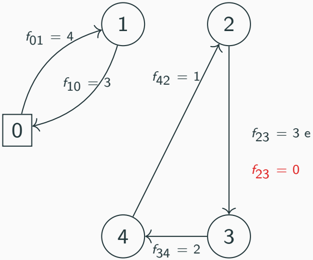
$$\begin{aligned}
\text{sujeito a: } & \sum_{i\in V} x_{ji} = 1 && \forall j \in V && \text{(um arco sai de j)} \\
& \sum_{i\in V} x_{ij} = 1 && \forall j \in V && \text{(um arco entra de j)} \\
& \sum_{i\in V\setminus\{0\}} f_{0i} = |V| - 1 && && \text{(sai } |V| - 1 \text{ fluxo de 0)} \\
& \sum_{j\in V: j\neq i} f_{ji} = \sum_{j\in V: j\neq i} f_{ij} + 1 && \forall i \in V \setminus \{0\} && \text{(fluxo entra } = \text{ fluxo sai } + \text{ fluxo fica)} \\
& 0 \leq f_{ij} \leq (|V| - 1) x_{ij} && \forall (i, j) \in N && \text{(limites var. fluxo)} \\
& x_{ij} \in \{0, 1\} && \forall (i, j) \in A &&
\end{aligned}$$

[3] Gavish, B., & Graves, S. C. (1978). *The travelling salesman problem and related problems*. Working paper GR-078-78, Operations Research Center, Massachusetts Institute of Technology.

Formulação SCF



(a) Solução admissível TSP.



(b) Solução não admissível TSP.

Figura 7: Como funciona o SCF.

Formulação SCF

- Consideremos $V = \{0, 1, 2\}$.

$$
\begin{aligned}
\min\ & c_{01}x_{01} + c_{02}x_{02} + c_{10}x_{10} + c_{12}x_{12} + c_{20}x_{20} + c_{21}x_{21} \\
\text{sujeito a: }\ & (\ldots) \\
& f_{01} + f_{02} = 2 \qquad \text{(sai 2 de fluxo do depósito)} \\
& f_{01} + f_{21} = f_{10} + f_{12} + 1 \qquad \text{(cons. fluxo 1)} \\
& f_{02} + f_{12} = f_{20} + f_{21} + 1 \qquad \text{(cons. fluxo 2)} \\
& 0 \leq f_{01} \leq 2x_{01} \qquad \text{(rel. f e x para (0,1))} \\
& 0 \leq f_{02} \leq 2x_{02} \qquad \text{(rel. f e x para (0,2))} \\
& 0 \leq f_{10} \leq 2x_{10} \qquad \text{(rel. f e x para (1,0))} \\
& 0 \leq f_{12} \leq 2x_{12} \qquad \text{(rel. f e x para (1,2))} \\
& 0 \leq f_{20} \leq 2x_{20} \qquad \text{(rel. f e x para (2,0))} \\
& 0 \leq f_{21} \leq 2x_{21} \qquad \text{(rel. f e x para (2,1))} \\
& x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A
\end{aligned}
$$

Formulação CC

- Formulação de cortes de conexidade - *Connectivity cuts (CC)*[4]

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\text{sujeito a: } \sum_{i\in V} x_{ji} = 1 \qquad \forall j \in V \qquad \text{(um arco sai de j)}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V \qquad \text{(um arco entra em j)}$$

$$\sum_{i\in V\setminus S} \sum_{j\in S} x_{ij} \geq 1 \qquad \forall S \subsetneq V : S \neq \emptyset \qquad \text{(um arco com início em S e fim em } V \setminus S \text{ tem que ser usado na solução)}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

[4] Dantzig, G., Fulkerson, R., & Johnson, S. (1954). *Solution of a large-scale traveling-salesman problem*. Journal of the Operations Research Society of America, 2(4), 393-410.

Formulação CC

- As restrições $\sum_{i \in V \setminus S} \sum_{j \in S} x_{ij} \geq 1$ são em número exponencial. $\Longrightarrow$ A formulação CC é não compacta.
  - Esta formulação deve ser resolvida com recurso a um algoritmo de branch-and-cut.
    - As restrições em número exponencial podem ser utilizadas como desigualdades válidas nas outras formulações apresentadas.

Formulação CC

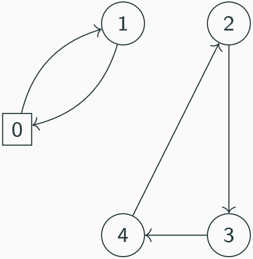


Figura 8: Solução não admissível para o TSP.

Considerando $S = \{2, 3, 4\}$, não existe nenhum arco com inicio em $V \setminus S = \{0, 1\}$ e com fim em S.

- Um corte para esta solução é $x_{02} + x_{03} + x_{04} + x_{12} + x_{13} + x_{14} \geq 1$.

Formulação CC

- Consideremos $V = \{0, 1, 2, 3\}$.

$$\begin{aligned}
\min\ & c_{01}x_{01} + c_{02}x_{02} + c_{03}x_{03} + c_{10}x_{10} + c_{12}x_{12} + c_{13}x_{13} + c_{20}x_{20} + \\
& c_{21}x_{21} + c_{23}x_{23} + c_{30}x_{30} + c_{31}x_{31} + c_{32}x_{32} \\
\text{sujeito a:}\ & (\ldots) \\
& x_{02} + x_{03} + x_{12} + x_{13} \geq 1 \\
& x_{01} + x_{03} + x_{21} + x_{23} \geq 1 \\
& x_{01} + x_{02} + x_{31} + x_{32} \geq 1 \\
& x_{10} + x_{13} + x_{20} + x_{23} \geq 1 \\
& x_{10} + x_{12} + x_{30} + x_{32} \geq 1 \\
& x_{20} + x_{21} + x_{30} + x_{31} \geq 1 \\
& x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A
\end{aligned}$$

- Que conjuntos S faltam?

Na formulação não estão presentes os conjuntos S de cardinalidade 1 e 3 porque são implicados pelas restrições de afetação.

Comparação da dimensão das formulações apresentadas

Tabela 1: Comparação de formulações em termos de número de variáveis e restrições.

Formulação	#Variáveis	#Restrições
MTZ	$\|A\| + \|V\| - 1$	$\|A\| + \|V\| - 1$
SCF	$2\|A\|$	$3\|V\| + \|A\|$
CC	$\|A\|$	$2^{\|V\|-1}$

- Qual das formulações é melhor?

Notação

- O problema de roteamento de veículos pode ser modelado num grafo orientado $G = (V, A)$, onde:
  - $V = \{0\} \cup N$.
    - 0 é o depósito; e
    - N é o conjunto dos clientes, que têm procura/oferta.
  - A é o conjunto dos arcos, que representam as ligações entre os clientes.
    - $A = \{(i, j) : i, j \in V \text{ e } i \neq j\}$
  - d_i é a procura do cliente $i \in N$.
    - D é a procura total, ou seja, $D = \sum_{i \in N} d_i$.
  - K é número de veículos homogéneos disponíveis.
  - Q é capacidade de cada veículo.
  - c_{ij} é o custo de atravessar o arco $(i, j) \in A$.
- Queremos determinar o conjunto de circuitos de custo total mínimo que: (i) comecem e acabem no depósito; (ii) satisfaçam a procura de todos os clientes em N; e (iii) a procura total de cada rota não exceda a capacidade dos veículos.

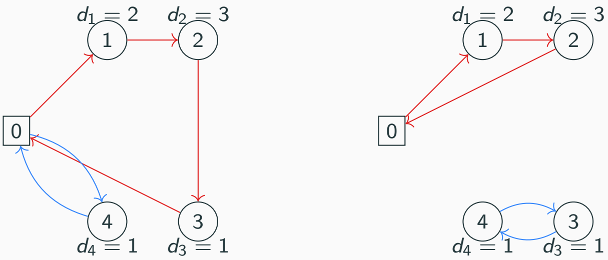
Formulação genérica

$$x_{ij} = \begin{cases} 1, \text{se o arco } (i,j) \in A \text{ é usado em alguma das rotas} \\ 0, \text{caso contrário} \end{cases}$$

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} && \\
\text{sujeito a:} \quad & \sum_{i \in N} x_{0i} = K && \text{(sai um arco do depósito para cada veículo)} \\
& \sum_{i \in V} x_{ji} = 1 && \forall j \in N \quad \text{(um arco sai do cliente j)} \\
& \sum_{i \in V} x_{ij} = 1 && \forall j \in N \quad \text{(um arco entra do cliente j)} \\
& \text{Não contém subcircuitos} && \\
& \text{Satisfaz a capacidade dos veículos} && \\
& x_{ij} \in \{0,1\} && \forall (i,j) \in A
\end{aligned}$$

Formulação genérica

- Considerando $K = 2$ e $Q = 4$.



(a) Excede capacidade veículos. **(b)** Contém subcircuitos.

Figura 9: Soluções não admissíveis para o VRP.

- Existem conjuntos de restrições que modelam as restrições de eliminação de subcircuitos e de capacidade.
  - Adaptações dos modelos de eliminação de subcircuitos para o TSP.

Formulação MTZ

- Formulação de Miller-Tucker-Zemlin (MTZ)

u_i - procura distribuída pelo veículo quando sai do nodo $i \in V \setminus \{0\}$.

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

sujeito a:

$$\sum_{i\in N} x_{0i} = K$$ (sai um arco do depósito para cada veículo)

$$\sum_{i\in V} x_{ji} = 1 \qquad \forall j \in N$$ (um arco sai do cliente j)

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in N$$ (um arco entra do cliente j)

$$u_i - u_j + Qx_{ij} \leq Q - d_j \qquad \forall (i,j) \in A : i,j \neq 0$$ (a procura distribuída com que o veículo sai de j é a com que saiu de i mais o que distribuiu em j, se foi de i para j)

$$d_i \leq u_i \leq Q \qquad \forall i \in N$$ (a procura distribuída quando o veículo sai de i está entre a procura de i e a capacidade veículo)

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

- Consideremos $K = 2$ e $Q = 4$.

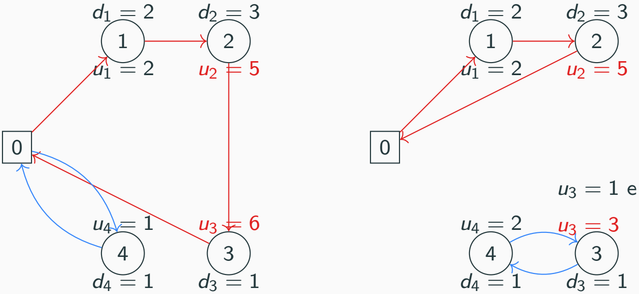


Figura 10: Soluções não admissíveis para o VRP.

- Considerando $K = 2$ e $Q = 4$.

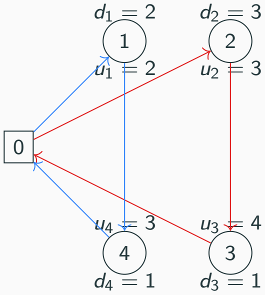


Figura 11: Solução admissível para o VRP.

Formulação SCF

- Formulação de fluxo único - Single Commodity Flow (SCF)

f_{ij} - fluxo que atravessa o arco $(i,j) \in A$, que corresponde à procura total por satisfazer quando o veículo atravessa arco $(i,j) \in A$.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

sujeito a:

$$\sum_{i\in N} x_{0i} = K \qquad \text{(sai um arco do depósito para cada veículo)}$$

$$\sum_{i\in V} x_{ji} = 1 \qquad \forall j \in N \quad \text{(um arco sai do cliente j)}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in N \quad \text{(um arco entra do cliente j)}$$

$$\sum_{i\in V\setminus\{0\}} f_{0i} = D \qquad \text{(D é a procura total por satisfazer quando os veículos saem do depósito)}$$

$$\sum_{j\in V:j\neq i} f_{ji} = \sum_{j\in V:j\neq i} f_{ij} + d_i \qquad \forall i \in N \quad \text{(procura por satisfazer quando entra em i é a com que sai de i mais a que foi satisfeita em i)}$$

$$0 \leq f_{ij} \leq Qx_{ij} \qquad \forall (i,j) \in A \quad \text{(limites capacidade disponível)}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

Formulação SCF

- Consideremos $K = 2$ e $Q = 4$.

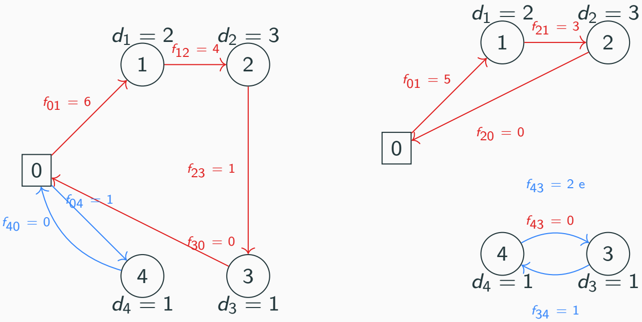


Figura 12: Soluções não admissíveis para o VRP.

Formulação SCF

- Consideremos $K = 2$ e $Q = 4$.

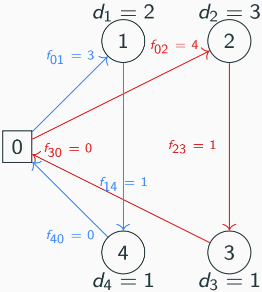


Figura 13: Solução admissível para o VRP.

Formulação SCF

- Os limites das variáveis f podem ser melhorados.
  - Quando saímos de i qual é a procura máxima por satisfazer?
    - Como o i já foi visitado, já lá ficaram d_i unidades de fluxo. Logo, a procura por satisfazer é menor ou igual que $Q - d_i$.
  - Quando entramos em j qual é a procura mínima por satisfazer?
    - Como o veículo ainda vai visitar o j, tem pelo menos a procura do j para satisfazer. Logo, a procura por satisfazer é maior ou igual que d_j.

- Podemos substituir as restrições limites capacidade disponível por

$$d_j x_{ij} \leq f_{ij} \leq (Q - d_i) x_{ij}, \qquad \forall (i,j) \in A$$

Nota: Para simplificar a apresentação das restrições consideramos que $d_0 = 0$.

Formulação RCC

- Formulação de cortes de capacidade arredondados - *Rounded capacity cuts* (RCC)[5]

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

sujeito a: $\sum_{i\in N} x_{0i} = K$ (sai um arco do depósito para cada veículo)

$$\sum_{i\in V} x_{ji} = 1 \qquad \forall j \in N$$ (um arco sai do cliente j)

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in N$$ (um arco entra do cliente j)

$$\sum_{i\in V\setminus S} \sum_{j\in S} x_{ij} \geq \left\lceil \frac{\sum_{i\in S} d_i}{Q} \right\rceil \qquad \forall S \subset N : S \neq \emptyset$$ (o número de veículos necessários para satisfazer a procura de um subconjunto de clientes é a soma da procura a dividir pela capacidade do veículo arredondada para cima [variáveis inteiras])

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

[5] Laporte, G., Mercure, H., & Nobert, Y. (1986). *An exact algorithm for the asymmetrical capacitated vehicle routing problem.* Networks, 16(1), 33-46.

Formulação RCC

- As restrições $\sum_{i \in V \setminus S} \sum_{j \in S} x_{ij} \geq \left\lceil \frac{\sum_{i \in S} d_i}{Q} \right\rceil$ são em número exponencial.
  - Esta formulação deve ser resolvida com recurso a um algoritmo de *branch-and-cut*.
    - As restrições em número exponencial podem ser utilizadas como desigualdades válidas para as outras formulações apresentadas

Formulação RCC

- Consideremos $K = 2$ e $Q = 4$.

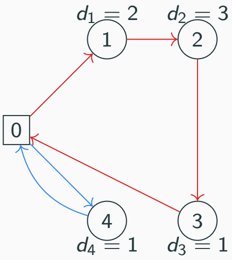


Figura 14: Solução não admissível para o VRP.

- Considere-se $S = \{1, 2, 3\}$. Temos $\sum_{i \in S} = d_1 + d_2 + d_3 = 2 + 3 + 1 = 6$ e $\lceil 6/4 \rceil = \lceil 1.50 \rceil = 2$ e apenas existe um arco com início em $V \setminus S$ e fim em S.
  - Um corte RCC para esta solução é $x_{01} + x_{02} + x_{03} + x_{41} + x_{42} + x_{43} \geq 2$.

Formulação RCC

- Consideremos $K = 2$ e $Q = 4$.

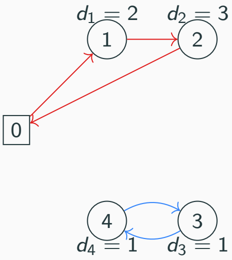


Figura 15: Solução não admissível para o VRP.

- Considere-se $S = \{3, 4\}$. Temos $\sum_{i \in S} = d_3 + d_4 = 1 + 1 = 2$ e $\lceil 2/4 \rceil = \lceil 0.50 \rceil = 1$ e não existe nenhum arco com início em $V \setminus S$ e fim em S usado na solução.
  - Um corte RCC para esta solução é $x_{03} + x_{04} + x_{13} + x_{14} + x_{23} + x_{24} \geq 1$.

Sumário

Relaxações do problema do caixeiro viajante

- Qualquer formulação válida para o problema do caixeiro garante que

  1. cada nodo tem um arco a entrar e um arco a sair; e
  2. não existem subcircuitos.

- Consideremos a formulação genérica:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{sujeito a:} \quad & \sum_{i\in V} x_{ji} = 1 & \forall j \in V \quad & \text{(um arco sai de j)} \\
& \sum_{i\in V} x_{ij} = 1 & \forall j \in V \quad & \text{(um arco entra j)} \\
& \text{Não contém subcircuitos} & & \\
& x_{ij} \in \{0,1\} & \forall (i,j) \in A &
\end{aligned}
$$

- Podemos obter relaxações removendo cada um dos conjuntos de restrições da formulação genérica.

Relaxações do problema do caixeiro viajante

- Relaxando as restrições de eliminação de subcircuitos obtemos:

$$
\begin{aligned}
(\text{REL1}) \equiv \min \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{sujeito a: } \sum_{i\in V} x_{ji} = 1 & \qquad \forall j \in V & \text{(um arco sai de j)} \\
\sum_{i\in V} x_{ij} = 1 & \qquad \forall j \in V & \text{(um arco entra j)} \\
x_{ij} \in \{0,1\} & \qquad \forall (i,j) \in A &
\end{aligned}
$$

- Que problema é este?

É o problema da afetação.

Relaxações do problema do caixeiro viajante

- Relaxando as restrições saída de um nodo obtemos:

$$
\begin{aligned}
(\text{REL2}) \equiv \min & \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{sujeito a: } & \sum_{i\in V} x_{ij} = 1 & & \forall j \in V \quad \text{(um arco entra j)} \\
& \text{Não contém subcircuitos} & & \\
& x_{ij} \in \{0,1\} & & \forall (i,j) \in A
\end{aligned}
$$

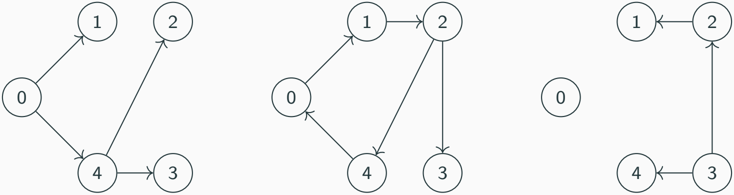
- Que problema é este?

Uma SA para a formulação apresentada anteriormente tem exatamente um arco a entrar em cada nodo e é conexa.

- Uma árvore é um grafo conexo e sem ciclos. Caso a árvore contenha todos os nodos do grafo original é uma árvore de suporte (ou árvore geradora).

- Uma arborescência é uma árvore com uma raiz - nodo sem "arcos a entrar" - e em que existe um único caminho da raiz para cada um dos restantes vértices da árvore. $\Longrightarrow$ Uma arborescência que contenha todos os nodos do grafo é uma arborescência geradora de G.

- Exemplos:



(a) Arborescência geradora de G. **(b)** Não arborescência. **(c)** Arborescência de G.

Relaxações do problema do caixeiro viajante

- Uma solução admissível da formulação REL2:

  1. contém todos os nodos do grafo,
  2. é conexa, e
  3. cada nodo tem exatamente um arco a entrar

- Se a uma arborescência geradora de G juntarmos um arco a entrar na raiz x, obtemos uma solução admissível para a REL2.

- Algoritmo para determinar minorantes baseado na REL2:

  Passo 1: Identificar em $G = (V, A)$ a arborescência de custo mínimo T com raiz em $x \in V$.

  Passo 2: Juntar a T o arco de menor custo a entrar em x. O valor de T é um minorante para o valor do TSP.

Relaxações do problema do caixeiro viajante

- O problema da afetação e o problema da arborescência de custo mínimo são problemas com um algoritmo polinomial conhecido. $\Longrightarrow$ "Problemas fáceis".

- Na prática, devem ser aplicados os algoritmos polinomiais para determinar a solução ótima dos problemas referidos.

- Como os algoritmos não estão no âmbito da disciplina, vamos resolver o problema da afetação e o problema da arborescência de custo mínimo utilizando modelos de programação linear inteira mista.

Relaxações do problema do roteamento de veículos

- Qualquer formulação válida para o problema do roteamento de veículos garante que

  1. cada nodo tem um arco a entrar e um arco a sair;
  2. não existem subcircuitos; e
  3. a procura de cada rota não excede a capacidade do veículo.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\
\text{sujeito a:} \quad & \sum_{i\in N} x_{0i} = K & & \text{(sai um arco do depósito para cada veículo)} \\
& \sum_{i\in V} x_{ji} = 1 & \forall j \in N \quad & \text{(um arco sai do cliente j)} \\
& \sum_{i\in V} x_{ij} = 1 & \forall j \in N \quad & \text{(um arco entra do cliente j)} \\
& \text{Não contém subcircuitos} & & \\
& \text{Satisfaz a capacidade dos veículos} & & \\
& x_{ij} \in \{0,1\} & \forall (i,j) \in A &
\end{aligned}
$$

Relaxações do problema de roteamento de veículos

- Podemos obter relaxações removendo cada conjunto de restrições da formulação genérica, contudo
  - Se mantivermos o conjunto de restrições Não contém subcircuitos obtemos o problema de do caixeiro viajante múltiplo, que é um problema difícil.
  - Se mantivermos o conjunto de restrições Satisfaz a capacidade de veículos obtemos o problema de empacotamento, que é um problema difícil.

- Apenas vamos considerar a relaxação em que relaxamos ambos os conjuntos de restrições simultaneamente.

Relaxações do problema de roteamento de veículos

- Relaxando as restrições de eliminação de subcircuitos e de capacidade de veículos obtemos:

$$(REL1) \equiv \min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{sujeito a: } \sum_{i\in N} x_{0i} = K \qquad \text{(sai um arco do depósito para cada veículo)}$$

$$\sum_{i\in V} x_{ji} = 1 \qquad \forall j \in N \quad \text{(um arco sai do cliente j)}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in N \quad \text{(um arco entra do cliente j)}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

Sumário

Heurística do vizinho mais próximo

Definição: Heurística do vizinho mais próximo

Começar numa cidade i^* e sucessivamente visitar a cidade mais próxima da última cidade visitada, terminado o circuito em i^*.

Heurística do vizinho mais próximo

```
Começar a rota R com uma cidade i*.
Fazer i = i*.
while Há cidades por visitar do
    Determinar a cidade j que ainda não está na rota que minimiza c_ij.
    Adicionar j ao fim da rota.
    Fazer i = j.
end while
```

- Seja $v(V + P)$ o valor da solução obtida com a heurística do vizinho mais próximo.

Exemplo

- Seja $R = \{A\}$ e $i = A$.
- Há cidades por visitar? Sim. Cidade j que minimiza c_{Aj} $\rightarrow$ Cidade E.
  - $R = (A, E)$ e $i = E$.

	A	B	C	D	E
A	-	3	5	19	2
B	3	-	14	15	11
C	5	14	-	20	1
D	19	15	20	-	9
E	2	11	1	9	-

- Há cidades por visitar? Sim. Cidade j que minimiza c_{Ej} $\rightarrow$ Cidade C.
  - $R = (A, E, C)$ e $i = C$.
- Há cidades por visitar? Sim. Cidade j que minimiza c_{Cj} $\rightarrow$ Cidade B.
  - $R = (A, E, C, B)$ e $i = B$.
- Há cidades por visitar? Sim. Cidade j que minimiza c_{Bj} $\rightarrow$ Cidade D.
  - $R = (A, E, C, B, D)$ e $i = D$.
- Há cidades por visitar? Não. Fim.

A rota obtida com a heurística do vizinho mais próximo é (A, E, C, B, D, A) e $v(V+P) = 2 + 1 + 14 + 15 + 19 = 51 \rightarrow gap = 54,55\%$.

Heurística de inserção de menor custo

Heurística de inserção de menor custo

1: Construir a rota R que contém as duas cidades i^* e j^*, tais que, $c_{i^*j^*} = min_{ij} c_{ij}$.
2: **while** Há cidades por visitar **do**
3: Determinar para cada cidade por visitar k e para cada par de cidades visitadas consecutivamente o menor custo de inserção, isto é, $c_{ik} + c_{jk} - c_{ij}$.
4: Inserir a cidade k entre as cidades i e j que minimize o custo de inserção.
5: **end while**

- Seja $v(IMC)$ o valor da solução obtida com a heurística de inserção mais próxima.

Exemplo

- Seja $R = \{\{C, E\}\}$.
- Há cidades por visitar?

  Sim. $R = \{\{C, A\}, \{A, E\}, \{E, C\}\}$
  - Cidade A: $c_{CA} + c_{AE} - c_{CE} = 5 + 2 - 1 = 4$. ← min
  - Cidade B: $c_{CB} + c_{BE} - c_{CE} = 14 + 11 - 1 = 24$.
  - Cidade D: $c_{CD} + c_{DE} - c_{CE} = 20 + 9 - 1 = 28$.

	A	*B*	*C*	*D*	*E*
A	-	3	5	19	2
B	3	-	14	15	11
C	5	14	-	20	1
D	19	15	20	-	9
E	2	11	1	9	-

- Há cidades por visitar? Sim. $R = \{\{C, B\}, \{B, A\}, \{A, E\}, \{E, C\}\}$
  - Cidade B:
    - Aresta $\{C, A\}$: $c_{CB} + c_{BA} - c_{CA} = 14 + 3 - 5 = 12$. ← min
    - Aresta $\{A, E\}$: $c_{AB} + c_{BE} - c_{AE} = 3 + 11 - 2 = 12$. ← min
    - Aresta $\{E, C\}$: $c_{EB} + c_{BC} - c_{EC} = 11 + 14 - 1 = 24$.
  - Cidade D:
    - Aresta $\{C, A\}$: $c_{CD} + c_{DA} - c_{CA} = 20 + 19 - 5 = 34$.
    - Aresta $\{A, E\}$: $c_{AD} + c_{DE} - c_{AE} = 19 + 9 - 2 = 26$.
    - Aresta $\{E, C\}$: $c_{ED} + c_{DC} - c_{EC} = 9 + 20 - 1 = 28$.

Exemplo (continuação)

- Há cidades por visitar? Sim. $R = \{\{C, D\}, \{D, B\}, \{B, A\}, \{A, E\}, \{E, C\}\}$
  - Cidade D:
    - Aresta $\{C, B\}$: $c_{CD} + c_{DB} - c_{CB} = 20 + 15 - 14 = 21$. $\leftarrow$ min
    - Aresta $\{B, A\}$: $c_{BD} + c_{DA} - c_{BA} = 15 + 19 - 3 = 31$.
    - Aresta $\{A, E\}$: $c_{AD} + c_{DE} - c_{AE} = 19 + 9 - 2 = 26$.
    - Aresta $\{E, C\}$: $c_{ED} + c_{DC} - c_{EC} = 9 + 20 - 1 = 28$.
- Há cidades por visitar? Não. Fim.

A rota obtida com a heurística de inserção de menor custo é (C, D, B, A, E, C) e $v(IMC) = 20 + 15 + 3 + 2 + 1 = 41 \rightarrow gap = 24,24\%$.

Heurísticas para o problema do roteamento de veículos

- Heurísticas construtivas
  - Heurística de Clarke & Wright.
- Heurísticas de decomposição
  - Agrupar $\rightarrow$ Rotear
    1. Particionar do conjunto de clientes N em K subconjuntos que satisfaçam as restrições de capacidade.
    2. Determinar uma rota admissível em cada um dos subconjuntos $\rightarrow$ Resolver um TSP.
  - Rotear $\rightarrow$ Agrupar
    1. Determinar uma rota que inclui todos os clientes $\rightarrow$ Resolver um TSP.
    2. Dividir a rota em K subrotas que satisfaçam as restrições de capacidade.

Heurística de Clarke & Wright

- É uma heurística gulosa para o problema do roteamento de veículos[6].

Ideia geral

Criar uma rota para cada nodo e ir juntando as rotas até obter uma solução admissível para o VRP.

[6] Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4), 568-581.

Heurística de Clarke & Wright

Heurística de Clarke & Wright

for all Cliente $i \in N$ **do**
  Calcular a rota $R_i = (0, i, 0)$.
end for
for all Par de clientes $i, j \in N : i \neq j$ **do**
  Calcular a poupança $s_{ij} = c_{i0} + c_{0j} - c_{ij}$.
end for
Fazer $R = \emptyset$ e $i \leftarrow 1$.
while A solução não é admissível **do**
  Seja (i^*, j^*) o par que origina a i-ésima maior poupança.
  if i^* e j^* não são visitados **then**
    Criar a nova rota $(0, i^*, j^*, 0)$ se $d_{i^*} + d_{j^*} < Q$ e eliminar R_{i^*} e R_{j^*}.
  else if Existem arcos $(0, i^*)$, $(0, j^*)$, $(i^*, 0)$ ou $(j^*, 0)$ em $R \cup \left(\cup_{i=1}^{|N|} R_i \right)$ **then**
    Juntar as rotas da seguinte forma $(0, \ldots, i^*, j^*, \ldots, 0)$ se a procura da nova rota não exceder Q.
  end if
  Fazer $i \leftarrow i + 1$.
end while

Exemplo

- Calcular a poupança para cada par de clientes.
  - Exemplo par $(1, 2)$: $s_{12} = c_{10} + c_{02} - c_{12} = 28 + 31 - 21 = 38$.

s_{ij}	A	B	C	D	E
A	-	38	19	27	42
B	-	-	13	36	33
C	-	-	-	15	27
D	-	-	-	-	34

Nota: Como a matriz de custos é simétrica temos $s_{ij} = s_{ji}$.

	0	A	B	C	D	E
0	-	28	31	20	25	34
A	28	-	21	29	26	20
B	31	21	-	38	20	32
C	20	29	38	-	30	27
D	25	26	20	30	-	25
E	34	20	32	27	25	-

$Q = 100$	A	B	C	D	E
d_i	37	35	30	25	32

- Ordenar pares de clientes pela maior poupança: (B, C), (C, D), (A, C), (A, D), (B, E), (D, E), (B, D), (A, B) e (A, E). Fazer $R = \emptyset$ e $i \leftarrow 1$.

- R é admissível? Não.
  - Considere-se o par (A, E).
  - Clientes A e E são não visitados? Sim.
    - Como $d_A + d_E = 69 < 100$, vamos criar uma nova rota. $\rightarrow$ $R = \{(0, A, E, 0)\}$.
  - $i \leftarrow i + 1 = 2$.

- R é admissível? Não.
  - Considere-se o par (A, B).
  - Clientes A e B não são visitados? Não.
  - Clientes A ou B estão adjacentes ao depósito no conjunto das rotas? Sim.
    - Como $d_B + d_A + d_E = 104 > 100$, não adicionamos um novo nodo à solução.
  - $i \leftarrow i + 1 = 3$.

Exemplo

- R é admissível? Não.
  - Considere-se o par (B, D).
  - Clientes B e D não são visitados? Sim.
    - Como $d_B + d_D = 60 \leq 100$, vamos criar uma nova rota. $\rightarrow R = \{(0, A, E, 0), (0, B, D, 0)\}$.
  - $i \leftarrow i + 1 = 4$.

	0	A	B	C	D	E
0	-	28	31	20	25	34
A	28	-	21	29	26	20
B	31	21	-	38	20	32
C	20	29	38	-	30	27
D	25	26	20	30	-	25
E	34	20	32	27	25	-

$Q = 100$	A	B	C	D	E
d_i	37	35	30	25	32

- R é admissível? Não.
  - Considere-se o par (D, E).
  - Clientes D e E não são visitados? Não.
  - Clientes D ou E estão adjacentes ao depósito no conjunto das rotas? Sim.
    - Como $d_A + d_E + d_B + d_D = 129 > 100$, não juntamos as rotas.

    .
  - $i \leftarrow i + 1 = 5$.
- R é admissível? Não.
  - Considere-se o par (B, E).
  - Clientes B e E não são visitados? Não.
  - Clientes B ou E estão adjacentes ao depósito no conjunto das rotas? Sim.
    - Como $d_B + d_C + d_D + d_E + d_A = 104 > 100$, não vamos juntar as rotas..
  - $i \leftarrow i + 1 = 6$.

Exemplo

- R é admissível? Não.
  - Considere-se o par (C, E) (empate).
  - Clientes C e E não são visitados? Não.
  - Clientes C ou E estão adjacentes ao depósito no conjunto das rotas? Sim.
    - Como $d_C + d_E + d_A \leq 100$, vamos adicionar C à rota $(0, A, E, 0)$. $\rightarrow R = \{(0, A, E, C, 0), (0, D, B, 0)\}$.
  - $i \leftarrow i + 1 = 7$.
- R é admissível? Sim. FIM.

	0	A	B	C	D	E
0	-	28	31	20	25	34
A	28	-	21	29	26	20
B	31	21	-	38	20	32
C	20	29	38	-	30	27
D	25	26	20	30	-	25
E	34	20	32	27	25	-

$Q = 100$	A	B	C	D	E
d_i	37	35	30	25	32

A solução obtida com a heurística do Clarke & Wright é $\{(0, A, E, C, 0); (0, B, D, 0)\}$, que tem valor $c_{0A} + c_{AE} + c_{EC} + c_{C0} + c_{0B} + c_{BD} + c_{D0} = 28 + 20 + 27 + 20 + 31 + 20 + 25 = 171$.

Exemplo

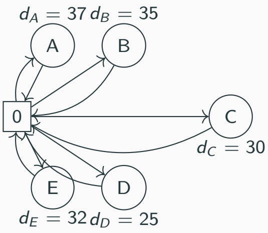


Figura 17: Inicialização.

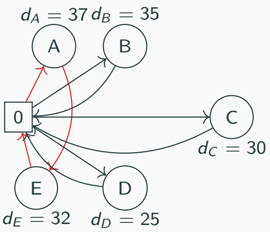


Figura 18: Iteração 1.

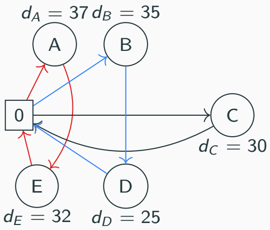


Figura 19: Iteração 3.

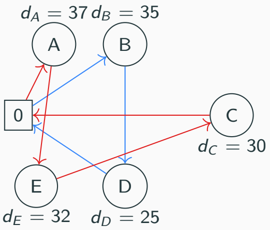


Figura 20: Iteração 7.

VRP Spreadsheet Solver

- O VRP Spreadsheet Solver é uma ferramenta desenvolvida em Visual Basic for Applications (VBA) que permite obter soluções admissíveis para diversas variantes do problema de roteamento de veículos.
  - Os procedimentos implementados no VRP Solver são heurísticos. $\Longrightarrow$ Não podemos garantir que a solução obtida pela ferramenta seja a ótima.

- O download da ferramenta pode ser feito em
  - `https://people.bath.ac.uk/ge277/vrp-spreadsheet-solver/`

- Detalhes sobre a ferramenta podem ser encontrados em
  - Erdoğan, G. (2017). *An open source spreadsheet solver for vehicle routing problems*. Computers & Operations Research, 84, 62-72.

Notas sobre o problema do caixeiro viajante

- O problema do caixeiro viajante é muito estudado na comunidade científica.
- Tem inúmeras variantes, como por exemplo:
  - Problema do caixeiro viajante com precedências.
  - Problema do caixeiro viajante com múltiplos depósitos.
  - Problema do caixeiro viajante com lucros.
  - Problema do comprador viajante.
  - Problema do caixeiro viajante com famílias.

Notas sobre o problema do roteamento de veículos

- Encontrar soluções admissíveis para o problema do roteamento de veículos pode ser um problema de difícil resolução dependendo da instância.
  - Resolver um problema de empacotamento.
- Existem inúmeras variantes do problema de roteamento de veículos, como por exemplo:
  - Problema do roteamento de veículos com janelas temporais.
  - Problema do roteamento de veículos com uma frota heterogénea.
  - Problema do roteamento de veículos com *backhauls*.
  - Problema do roteamento de veículos periódico.
  - Problema do roteamento de veículos com *split delivery*.

Referências bibliográficas

- Drexl, M. (2012). *Rich Vehicle Routing in Theory and Practice, Logistics Research*, Volume 5, pp. 47-63 (DOI: 10.1007/s12159-012-0080-2).
- Erdoğan, G. (2017). *An open source spreadsheet solver for vehicle routing problems*. Computers & Operations Research, 84, 62-72.
- Gutin, G., & Punnen, A. P. (Eds.). (2006). *The Traveling Salesman Problem and its Variations*, Springer Science & Business Media, New York.
- Toth, P. & D. Vigo (2014). *Vehicle Routing Problems, Methods, and Application*, 2nd ed., MOS-SIAM Series on Optimization, Philadelphia.
- Wolsey, L. A. (1998). *Integer programming*. John Wiley & Sons.