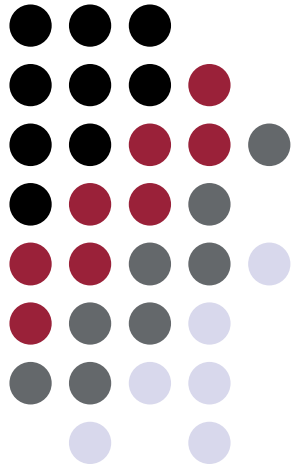
A03 Heurísticas Construtivas

PEP300 Técnicas Metaheuristicas para Otimização Combinatória



Prof. Dr. George H. G. Fonseca
Universidade Federal de Ouro Preto

Introdução

- Dois tipos de heurísticas
  - Construtivas – constroem uma solução passo a passo, elemento por elemento
  - Refinamento – consistem em melhorar uma solução através de modificações em seus elementos

Heurística de Construção Gulosa

- Constrói uma solução elemento por elemento
- A cada passo é adicionado um único elemento candidato
- O candidato escolhido é o melhor segundo algum critério
- O método encerra quando todos os elementos candidatos foram analisados

Heurística de Construção Gulosa

procedimento $ConstrucaoGulosa(g(.), s)$;
1 $s \leftarrow \emptyset$;
2 Inicialize o conjunto C de elementos candidatos;
3 enquanto $(C \neq \emptyset)$ faça
4 $\quad g(t_{melhor}) = melhor\{g(t) \mid t \in C\}$;
5 $\quad s \leftarrow s \cup \{t_{melhor}\}$;
6 $\quad$ Atualize o conjunto C de elementos candidatos;
7 fim-enquanto;
8 Retorne s;
fim $ConstrucaoGulosa$;

Heurística de Construção Gulosa

- Problema da mochila
  - 1º passo: calcular a relação benefício / peso

Objeto	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Benefício	4	3	2	6	2	3	5	4
Peso	5	4	3	9	4	2	6	7
Benefício / peso	0,80	0,75	0,67	0,67	0,50	1,50	0,83	0,57

Capacidade: 20

Heurística de Construção Gulosa

- Problema da mochila
  - 2º passo: ordenar os objetos

Objeto	**6**	**7**	**1**	**2**	**3**	**4**	**8**	**5**
Benefício	3	5	4	3	2	6	4	2
Peso	2	6	5	4	3	9	7	4
Benefício / peso	1,50	0,83	0,80	0,75	0,67	0,67	0,57	0,50

Capacidade: 20

Heurística de Construção Gulosa

- Problema da mochila
  - 3º passo: escolher o elemento de maior benefício / peso que respeite a capacidade da mochila até que não seja possível mais adicionar elementos

Objeto	**6**	**7**	**1**	**2**	**3**	**4**	**8**	**5**
Benefício	3	5	4	3	2	6	4	2
Peso	2	6	5	4	3	9	7	4
Benefício / peso	1,50	0,83	0,80	0,75	0,67	0,67	0,57	0,50

Mochila: {6}
Benefício: 3
Peso: 2
Capacidade: 20

Heurística de Construção Gulosa

- Problema da mochila
  - 3º passo: escolher o elemento de maior benefício / peso que respeite a capacidade da mochila até que não seja possível mais adicionar elementos

Objeto	**6**	**7**	**1**	**2**	**3**	**4**	**8**	**5**
Benefício	3	5	4	3	2	6	4	2
Peso	2	6	5	4	3	9	7	4
Benefício / peso	1,50	0,83	0,80	0,75	0,67	0,67	0,57	0,50

Mochila: {6, 7}
Benefício: 8
Peso: 8
Capacidade: 20

Heurística de Construção Gulosa

- Problema da mochila
  - 3º passo: escolher o elemento de maior benefício / peso que respeite a capacidade da mochila até que não seja possível mais adicionar elementos

Objeto	**6**	**7**	**1**	**2**	**3**	**4**	**8**	**5**
Benefício	3	5	4	3	2	6	4	2
Peso	2	6	5	4	3	9	7	4
Benefício / peso	1,50	0,83	0,80	0,75	0,67	0,67	0,57	0,50

Mochila: {6, 7, 1}
Benefício: 12
Peso: 13
Capacidade: 20

Heurística de Construção Gulosa

- Problema da mochila
  - 3º passo: escolher o elemento de maior benefício / peso que respeite a capacidade da mochila até que não seja possível mais adicionar elementos

Objeto	**6**	**7**	**1**	**2**	**3**	**4**	**8**	**5**
Benefício	3	5	4	3	2	6	4	2
Peso	2	6	5	4	3	9	7	4
Benefício / peso	1,50	0,83	0,80	0,75	0,67	0,67	0,57	0,50

Mochila: {6, 7, 1, 2}
Benefício: 15
Peso: 17
Capacidade: 20

Heurística de Construção Gulosa

- Problema da mochila
  - 3º passo: escolher o elemento de maior benefício / peso que respeite a capacidade da mochila até que não seja possível mais adicionar elementos

Objeto	**6**	**7**	**1**	**2**	**3**	**4**	**8**	**5**
Benefício	3	5	4	3	2	6	4	2
Peso	2	6	5	4	3	9	7	4
Benefício / peso	1,50	0,83	0,80	0,75	0,67	0,67	0,57	0,50

Mochila: {6, 7, 1, 2, 3}
Benefício: 17
Peso: 20
Capacidade: 20

Heurística de Construção Gulosa

- Problema da mochila
  - 3º passo: escolher o elemento de maior benefício / peso que respeite a capacidade da mochila até que não seja possível mais adicionar elementos

Objeto	**6**	**7**	**1**	**2**	**3**	**4**	**8**	**5**
Benefício	3	5	4	3	2	6	4	2
Peso	2	6	5	4	3	9	7	4
Benefício / peso	1,50	0,83	0,80	0,75	0,67	0,67	0,57	0,50

Mochila: {6, 7, 1, 2, 3}
Benefício: 17
Peso: 20
Capacidade: 20

Não é possível adicionar elementos!

Heurística de Construção Gulosa

procedimento *ConstrucaoGulosa*$(g(.), s)$;
1 $s \leftarrow \emptyset$;
2 Inicialize o conjunto C de elementos candidatos;
3 enquanto $(C \neq \emptyset)$ faça
4 $g(t_{max}) = \max\{g(t) \mid t \in C\}$;
5 $s \leftarrow s \cup \{t_{max}\}$;
6 Atualize o conjunto C de elementos candidatos;
7 fim-enquanto;
8 Retorne s;
fim *ConstrucaoGulosa*;

Heurística de Construção Gulosa

- Problema do caixeiro viajante
  - Parâmetros:
    C: conjunto de cidades
    d_{ij}: distância da cidade *i* à cidade *j*

  - Variáveis de decisão:

$$x_{ij} = \begin{cases} 1\ caso\ a\ aresta\ (i,j) seja\ usada \\ 0\ caso\ contrário \end{cases}$$

Problema do caixeiro viajante

- Parâmetros:
  C: conjunto de cidades
  d_{ij}: distância da cidade *i* à cidade *j*

- Variáveis de decisão:

$$x_{ij} = \begin{cases} 1\ caso\ a\ aresta\ (i,j)\ seja\ usada \\ 0\ caso\ contrário \end{cases}$$

$$f_{ij} = quantidade\ de\ fluxo\ de\ i\ para\ j$$

Problema do caixeiro viajante

- Função objetivo:

$$\min \sum_{i \in C} \sum_{j \in C} d_{ij} x_{ij}$$

- Restrições:

  1. De cada cidade *i* só sai uma aresta

$$\sum_{j \in C} x_{ij} = 1 \qquad \forall\, i \in C$$

  2. A cada cidade *j* só chega uma aresta

$$\sum_{i \in C} x_{ij} = 1 \qquad \forall\, j \in C$$

Problema do caixeiro viajante

- Restrições:

  3. O fluxo que chega a uma cidade i menos o que sai é igual a uma unidade

$$\sum_{j \in C} f_{ji} - \sum_{j \in C} f_{ij} = 1 \qquad \forall\, i \in C, i \neq 1$$

  4. O fluxo máximo em cada aresta é igual a *n*-1, onde *n* é o número de cidades

$$f_{ij} \leq (n-1)x_{ij} \qquad \forall\, i \in C, \forall\, j \in C$$

  5. Restrições de domínio

$$x_{ij} \in \{0,1\} \qquad \forall\, i \in C, \forall\, j \in C$$

$$f_{ij} \geq 0 \qquad \forall\, i \in C, \forall\, j \in C$$

Heurísticas construtivas para o Problema do Caixiero Viajante

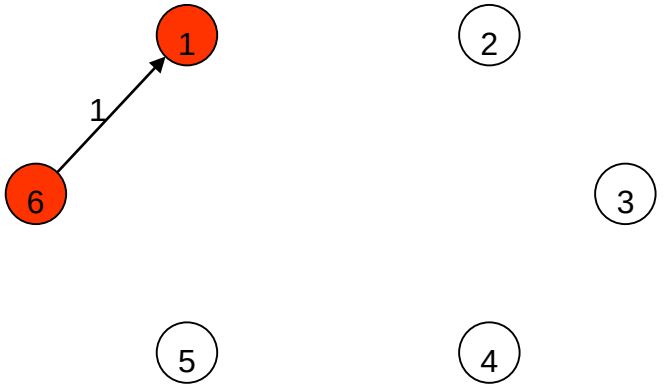
- Vizinho mais próximo
  - Ideia central: construir uma rota passo a passo, adicionando à solução corrente a cidade mais próxima (e ainda não visitada) da última cidade inserida

- Inserção mais barata
  - Ideia central: construir uma rota passo a passo, partindo de rota inicial envolvendo 3 cidades (obtidas por um método qualquer) e adicionar a cada passo, a cidade k (ainda não visitada) entre a ligação (i, j) de cidades já visitadas, cujo custo de inserção seja o mais barato

Vizinho mais próximo para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
6	1	1
6	2	2
6	3	6
6	4	6
6	5	2

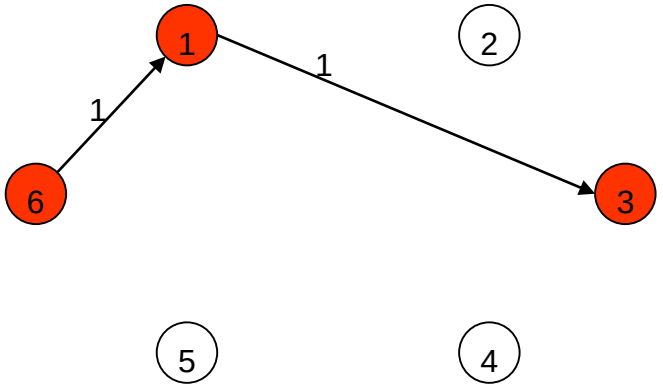


Distância total: 1

Vizinho mais próximo para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
1	2	2
1	3	1
1	4	4
1	5	9

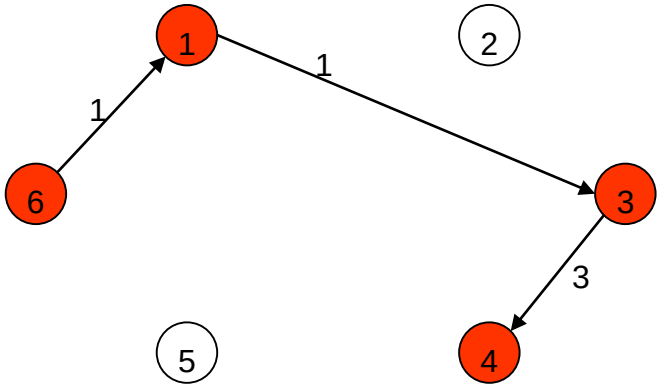


Distância total: 1 + 1 = 2

Vizinho mais próximo para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
3	2	5
3	4	3
3	5	8

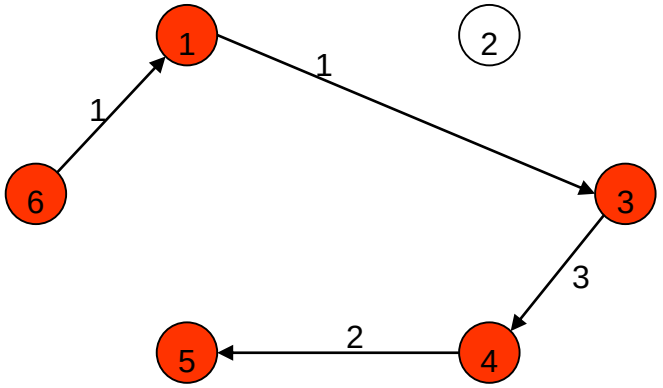


Distância total: 2 + 3 = 5

Vizinho mais próximo para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	D_{ij}
4	2	9
4	5	2

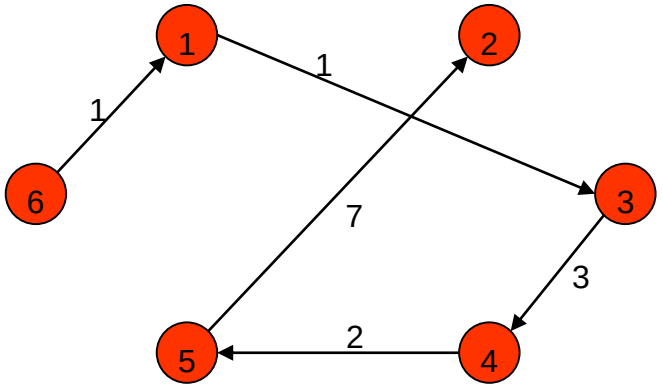


Distância total: 5 + 2 = 7

Vizinho mais próximo para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	D_{ij}
5	2	7

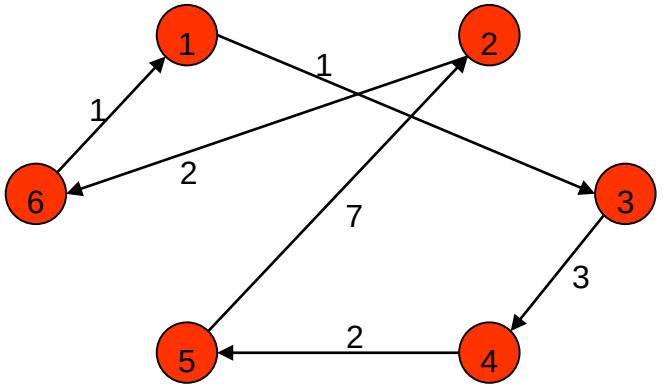


Distância total: 7 + 7 = 14

Vizinho mais próximo para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

Inserção forçada para fechar o ciclo



Distância total: 14 + 2 = 16

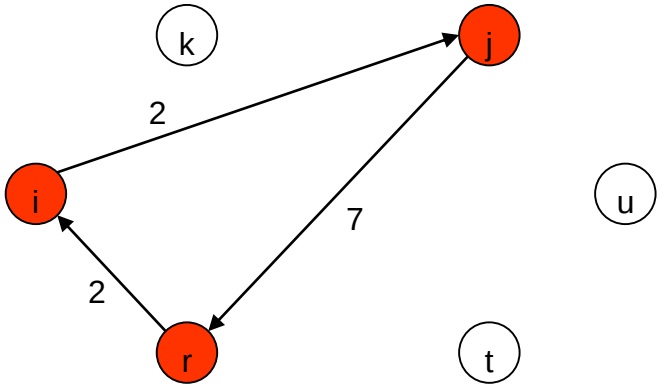
Vizinho mais próximo para o Problema do Caixiero Viajante

- Complexidade

Iteração	**Núm. de avaliações**
1	N – 1
2	N – 2
…	…
N – 1	1
N	1
Total	1 + N(N - 1) / 2

Inserção mais barata para o Problema do Caixiero Viajante

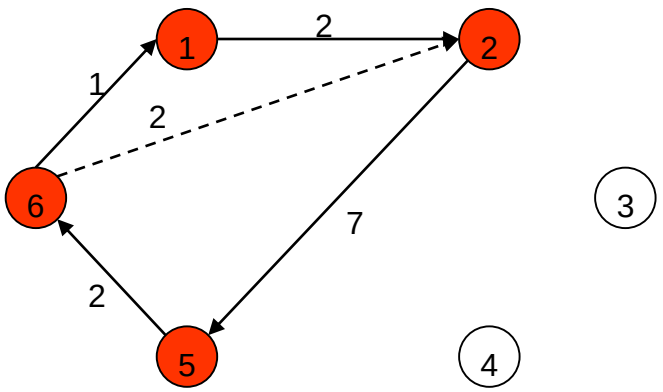
Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0



Custo da inserção de k entre i e j: $d_{ik} + d_{kj} - d_{ij}$

Inserção mais barata para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0



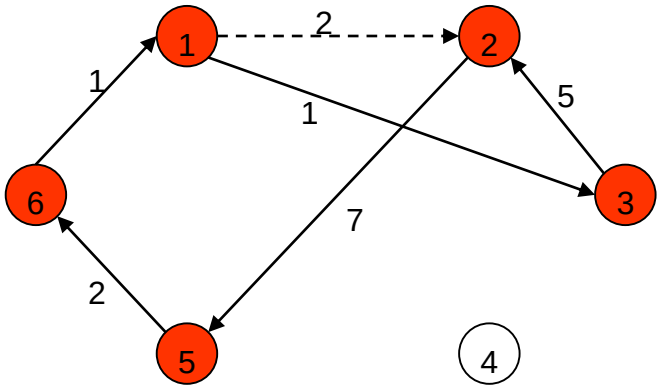
i	*k*	*j*	$d_{ik} + d_{kj} - d_{ij}$
6	1	2	1 + 2 – 2 = 1
6	3	2	6 + 5 – 2 = 9
6	4	2	6 + 9 – 2 = 13
2	1	5	2 + 9 – 7 = 4
2	3	5	5 + 8 – 7 = 6
2	4	5	9 + 2 – 7 = 4
5	1	6	9 + 1 – 2 = 8
5	3	6	8 + 6 – 2 = 12
5	4	6	2 + 6 – 2 = 6

Distância total: 12

Inserção mais barata para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*k*	*j*	$d_{ik} + d_{kj} - d_{ij}$
6	3	1	6 + 1 – 1 = 6
6	4	1	6 + 4 – 1 = 9
1	3	2	1 + 5 – 2 = 4
1	4	2	4 + 9 – 2 = 11
2	3	5	5 + 8 – 7 = 6
2	4	5	9 + 2 – 7 = 4
5	3	6	8 + 6 – 2 = 12
5	4	6	2 + 6 – 2 = 6

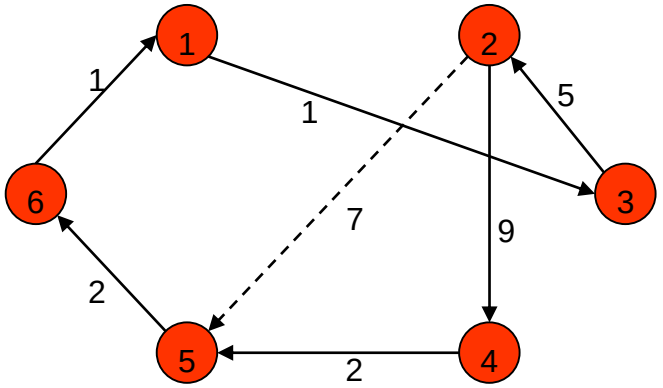


Distância total: 16

Inserção mais barata para o Problema do Caixiero Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*k*	*j*	$d_{ik} + d_{kj} - d_{ij}$
6	4	1	6 + 4 – 1 = 9
1	4	3	4 + 3 – 1 = 6
3	4	2	3 + 9 – 5 = 7
2	4	5	9 + 2 – 7 = 4
5	4	6	2 + 6 – 2 = 6



Distância total: 20

Vizinho mais próximo para o Problema do Caixiero Viajante

- Complexidade

Iteração	**Núm. de avaliações**
1	3(N – 3)
2	4(N – 4)
…	…
i – 2	i(N - 1)
…	…
N - 3	(N - 1)(N - (N - 1))
Total	$\sum_{i=3}^{N-1} i(N-1)$

$$\sum_{i=3}^{n-1} i(n-i) = \frac{1}{6}n^3 - n^2 - \frac{5}{6}n - 3$$

Heurística construtiva parcialmente gulosa

- Invés de inserir sempre o melhor candidato na solução a cada passo, sorteia um dentre os melhores n candidatos

- A lista dos n candidatos é conhecida como lista de candidatos restrita (LCR)

- Um parâmetro α = [0, 1] define a aleatoriedade/ gulosidade do método
  - α mais próximo de 0 – heurística gulosa
  - α mais próximo de 1 – heurística aleatória

Heurística construtiva parcialmente gulosa

procedimento $Construcao(g(.), \alpha, s)$;
1 $s \leftarrow \emptyset$;
2 Inicialize o conjunto $\mathcal{C}$ de candidatos;
3 enquanto $(\mathcal{C} \neq \emptyset)$ faça
4 $g(t_{min}) = \min\{g(t) \mid t \in \mathcal{C}\}$;
5 $g(t_{max}) = \max\{g(t) \mid t \in \mathcal{C}\}$;
6 $LCR = \{t \in \mathcal{C} \mid g(t) \leq g(t_{min}) + \alpha(g(t_{max}) - g(t_{min}))\}$;
7 Selecione, aleatoriamente, um elemento $t \in LCR$;
8 $s \leftarrow s \cup \{t\}$;
9 Atualize o conjunto $\mathcal{C}$ de candidatos;
10 fim-enquanto;
11 Retorne s;
fim $Construcao$;

Heurística construtiva parcialmente gulosa

- Problema do caixeiro viajante
  - Ordenar as cidades não vizitadas da mais próxima para a mais distante da atual
  - Definir o tamanho da lista restrita de candidatos (LRC)

$$[\text{tamLRC} = \text{tamMin} + \alpha(\text{tamMax} - \text{tamMin})]$$

tamLRC = 1 + 0(8 - 1) = 1
tamLRC = 1 + 0.3(8 - 1) = 3,1 = 4
tamLRC = 1 + 0.5(8 - 1) = 4,5 = 5
tamLRC = 1 + 1(8 - 1) = 8

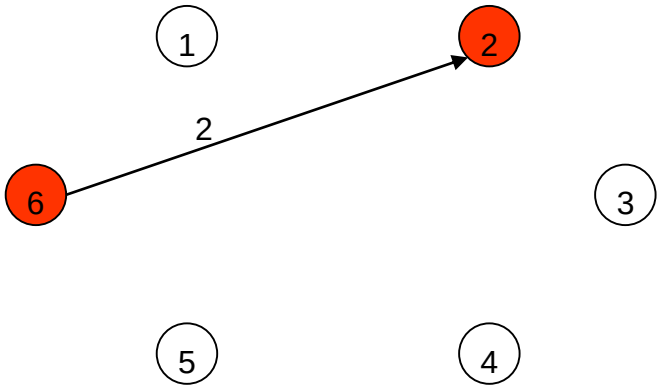
  - A cada elemento adicionado, recalcular o tamanho da LRC

Heurística construtiva parcialmente gulosa para o PCV

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
6	1	1
6	**2**	**2**
6	5	2
6	3	6
6	4	6

Sorteado



Supondo $\alpha = 0,3$

tamLRC = 1 + 0,3(5 - 1) = 2,2 → 3

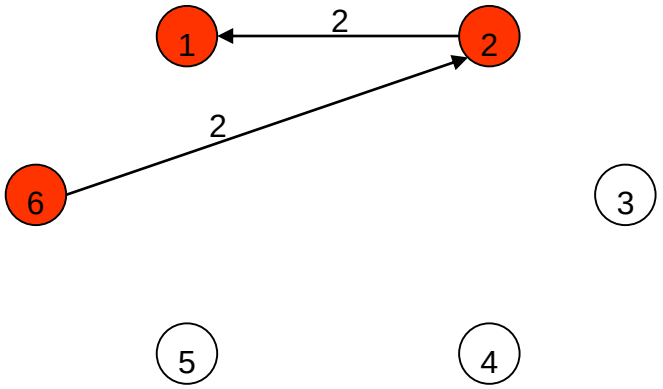
Distância total: 2

Heurística construtiva parcialmente gulosa para o PCV

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
2	**1**	**2**
2	3	5
2	5	7
2	4	9

Sorteado



Supondo $\alpha = 0,3$

tamLRC = 1 + 0,3(4 - 1) = 1,9 → 2

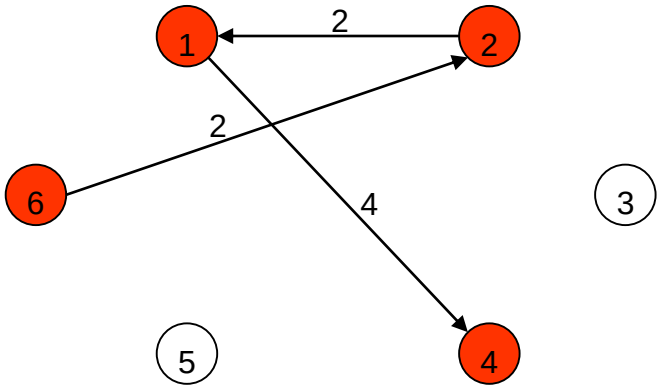
Distância total: 4

Heurística construtiva parcialmente gulosa para o PCV

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
1	3	1
1	**4**	**4**
1	5	9

Sorteado



Supondo $\alpha = 0,3$

tamLRC = 1 + 0,3(3 - 1) = 1,6 → 2

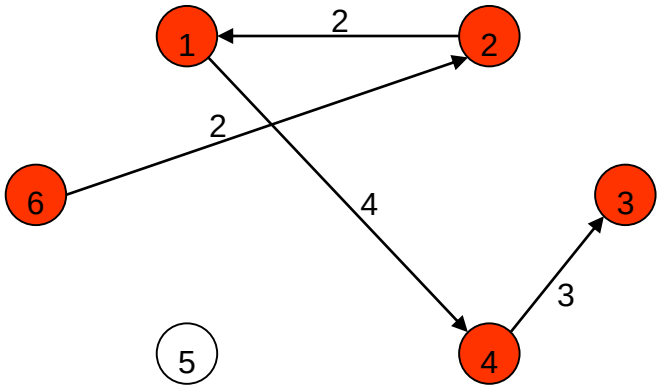
Distância total: 8

Heurística construtiva parcialmente gulosa para o PCV

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
4	5	2
4	**3**	**3**

Sorteado



Supondo $\alpha = 0,3$

tamLRC = 1 + 0,3(2 - 1) = 1,3 → 2

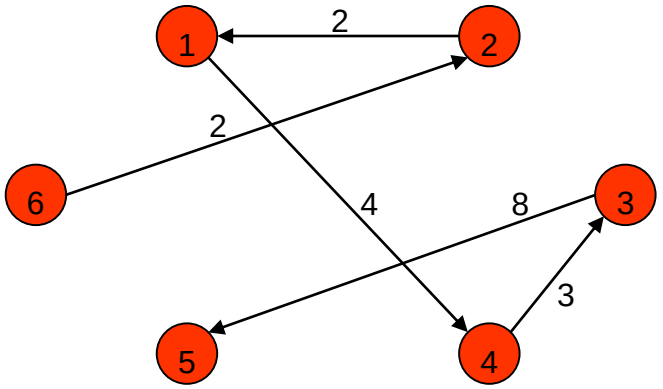
Distância total: 11

Heurística construtiva parcialmente gulosa para o PCV

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

i	*j*	d_{ij}
3	**5**	**8**

Sorteado



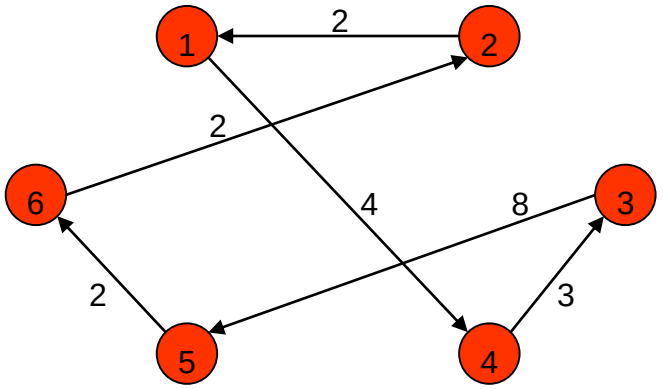
Supondo α = 0,3
tamLRC = 1 + 0,3(1 - 1) = 1

Distância total: 19

Heurística construtiva parcialmente gulosa para o PCV

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

Inserção forçada para fechar o ciclo



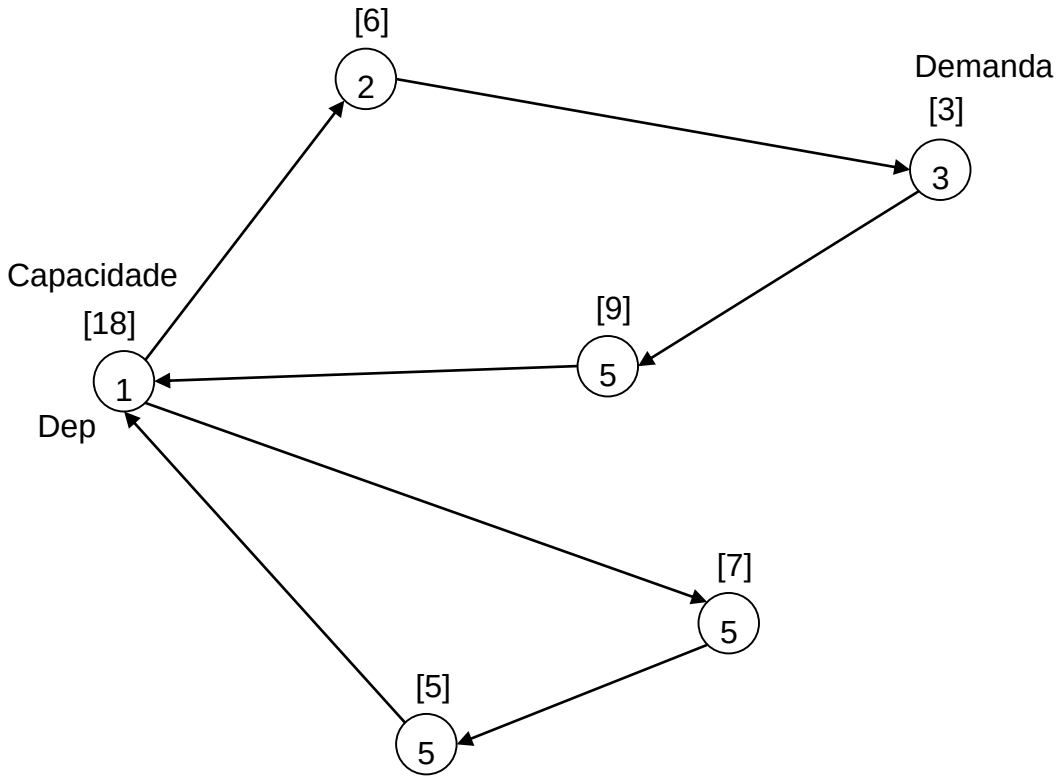
Distância total: 21

Problema do Roteamento de Veículos

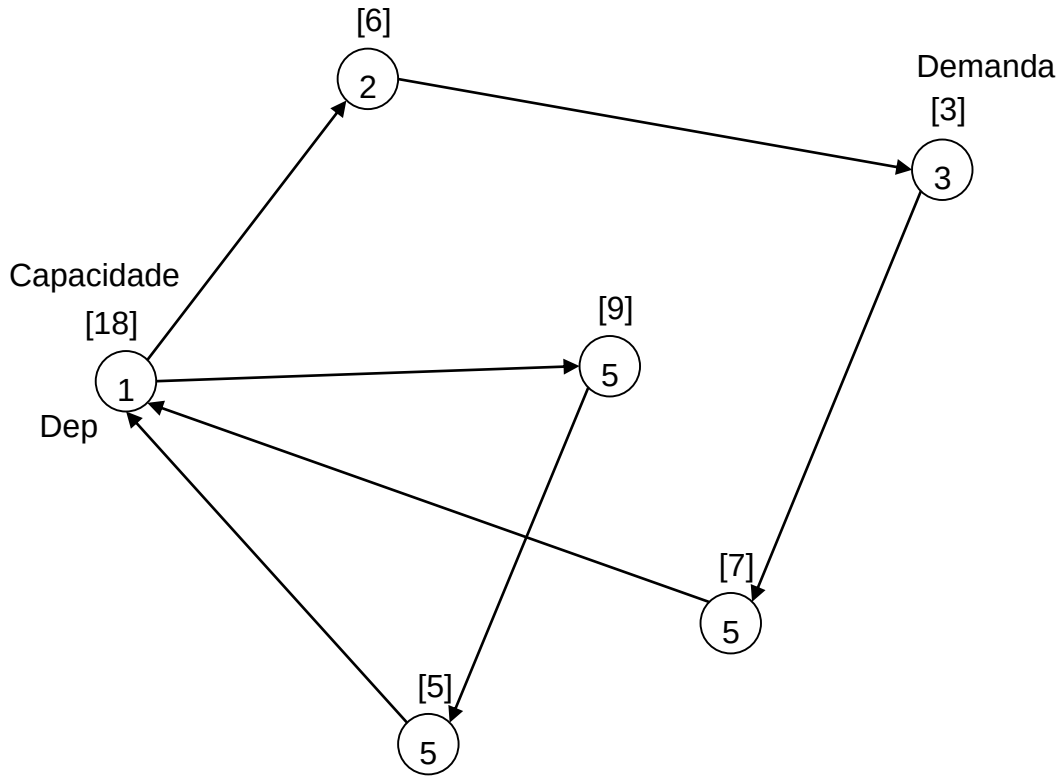
- Dados de entrada
  - Um depósito
  - Uma frota de veículos, com base no depósito
  - Um conjunto de clients
  - A demanda de cada cliente
  - Uma matriz de distâncias D = d_{ij} entre depósito e clientes e entre pares de clientes
  - Cidades = depósito U clientes

- PRV consiste em encontrar um conjunto de rotas para os veículos tal que:
  - Cada rota comece e termine no depósito
  - Cada cliente seja atendido por um único veículo
  - A capacidade dos veículos seja respeitada
  - A distância total percorrida seja a menor possível

Problema do Roteamento de Veículos



Problema do Roteamento de Veículos



Formulação Matemática para o PRV

- Dados de entrada
  - *C*: conjunto formado por um depósito (1) e clientes
  - d_{ij}: distância entre as cidades *i* e *j*
  - dem_i: demanda da cidade *i*
  - *cap*: capacidade dos veículos

- Variáveis de decisão
  - x_{ij}: 1 se a aresta (*i, j*) será usada; 0 caso contrário
  - f_{ij}: quantidade de fluxo de *i* para *j*

- Função objetivo

$$\min \sum_{i \in C} \sum_{j \in C} d_{ij} x_{ij}$$

Formulação Matemática para o PRV

- Restrições
  - De cada cidade i, exceto o depósito (1), só sai um veículo

$$\sum_{j \in C} x_{ij} = 1 \qquad \forall\, i \ \in C\,, i \neq 1$$

  - A cada cidade j, exceto o depósito, só chega um veículo

$$\sum_{i \in C} x_{ij} = 1 \qquad \forall\, j \ \in C, j \neq 1$$

  - O número de veículos que saem do depósito é igual ao que chegam ao depósito

$$\sum_{j \in C} x_{1j} = \sum_{i \in C} x_{i1}$$

Formulação Matemática para o PRV

- Restrições
  - Ao passar por uma cidade j, exceto o depósito, o veículo deve atender a demanda dessa cidade, isto é, deve deixar dem_j unidades de produto na cidade j

$$\sum_{i \in C} f_{ij} - \sum_{i \in C} f_{ji} = dem_i \qquad \forall\, j \in C$$

  - O fluxo máximo em cada aresta não pode superar a cap. do veículo

$$f_{ij} \leq cap\, x_{ij} \qquad \forall\, i \in C, \forall\, j \in C$$

  - Domínio das variáveis

$$x_{ij} \in \{0, 1\} \qquad \forall\, i \in C, \forall\, j \in C$$

$$f_{ij} \geq 0 \qquad \forall\, i \in C, \forall\, j \in C$$

Adaptação da heurística do vizinho mais próximo para o PRV

- Ideia principal
  - Passo 1: partir de um depósito com um novo veículo e ir até a cidade mais próxima ainda não visitada
  - Passo 2: determinar a cidade mais próxima da última cidade inserida na rota e verificar se é possível atender sua demanda
  - Passo 3: Se for possível atender a demanda dessa cidade, adicioná-la à rota. Caso contrário, retornar ao depósito e voltar ao passo 1

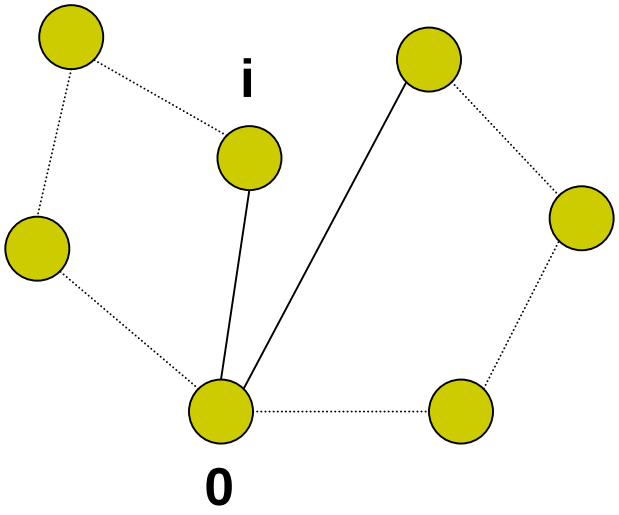
Heurística Construtiva de Clark & Wright para o PRV

- Ideia principal
  - Colocar um veículo atendendo cada cliente, isto é, considerar n veículos saindo do depósito, atendendo cada qual a um único cliente e retornando ao depósito;
  - Unir as rotas de cada veículo com base no conceito de economia

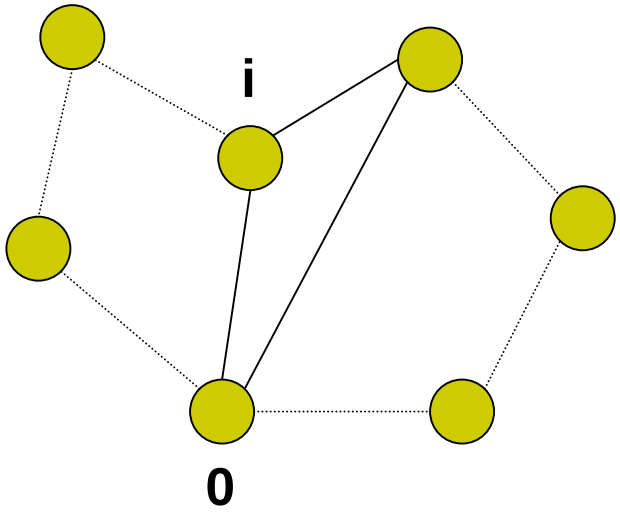
- À medida que se reduz a distância total percorrida, o número de veículos necessários também é reduzido

Heurística Construtiva de Clark & Wright para o PRV

Rota inicial



Rota combinada

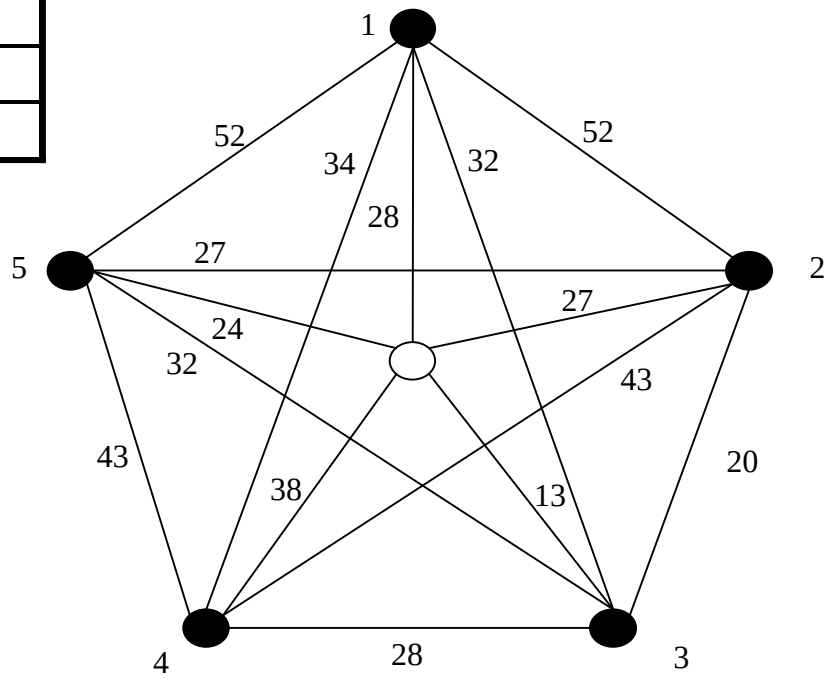


Economia $s_{ij} = d_{i0} + d_{0j} - d_{ij}$

i e j devem ser clientes das extremidades das rotas

Cidades	1	2	3	4	5	CAP
Demanda	15	17	27	12	23	50

i	j	d_{i0}	d_{i0}	d_{ij}	S_{ij}	Dem

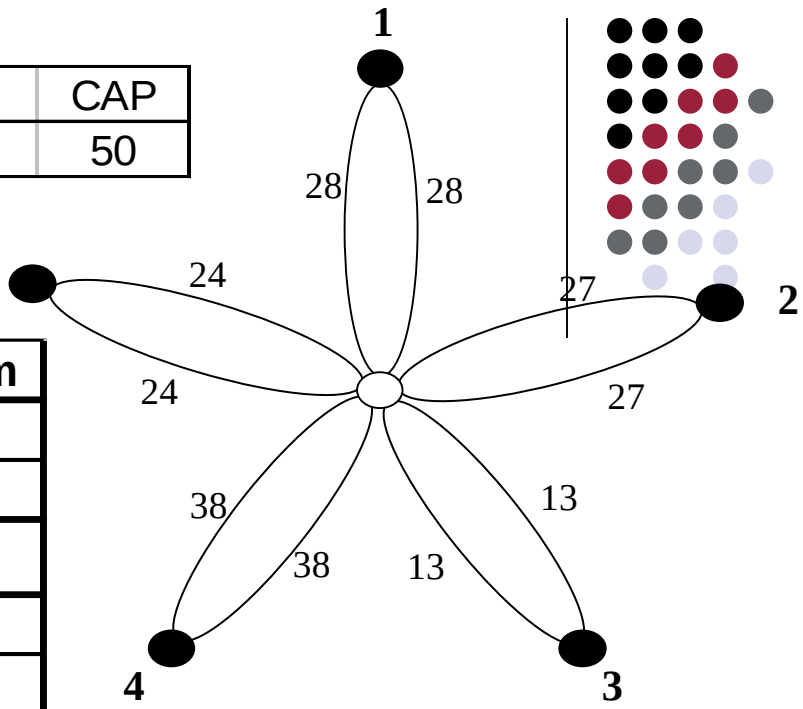


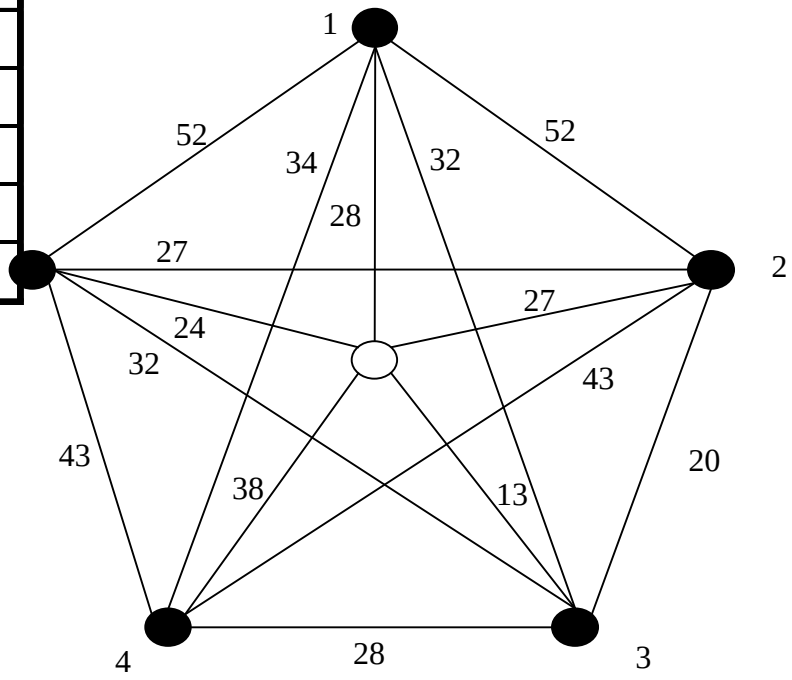
$$s_{ij} = d_{i0} + d_{j0} - d_{ij}$$

Cidades	1	2	3	4	5	CAP
Demanda	15	17	27	12	23	50

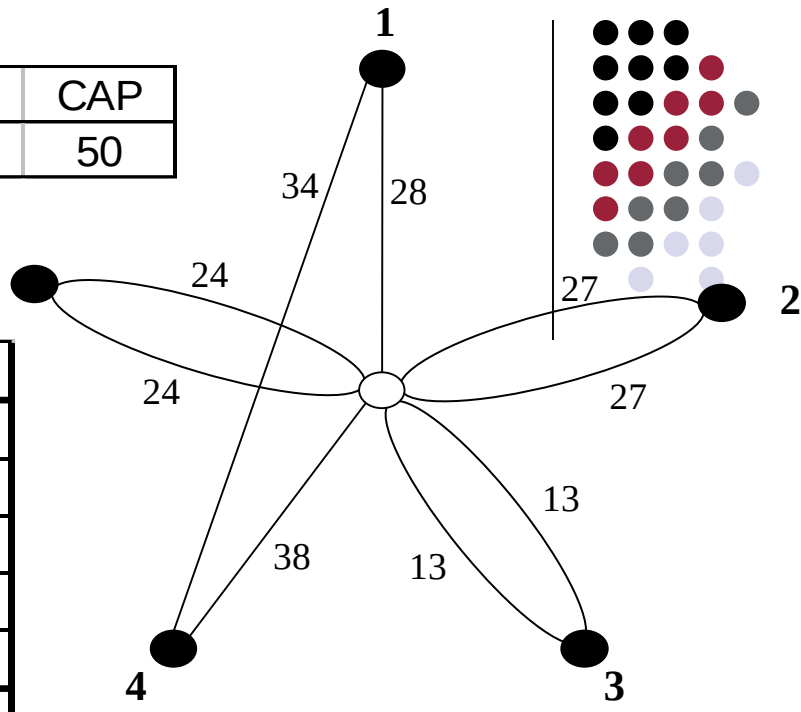
i	j	d_{i0}	d_{j0}	d_{ij}	S_{ij}	Dem
1	2	28	27	52	3	32
1	3	28	13	32	9	42
1	**4**	**28**	**38**	**34**	**32**	**27**
1	5	28	24	52	0	38
2	3	27	13	20	20	44
2	4	27	38	43	22	29
2	5	27	24	27	24	40
3	4	13	38	28	23	39
3	5	13	24	32	5	50
4	5	38	24	43	19	35^5

Total percorrido:	260
Nº de caminhões:	5





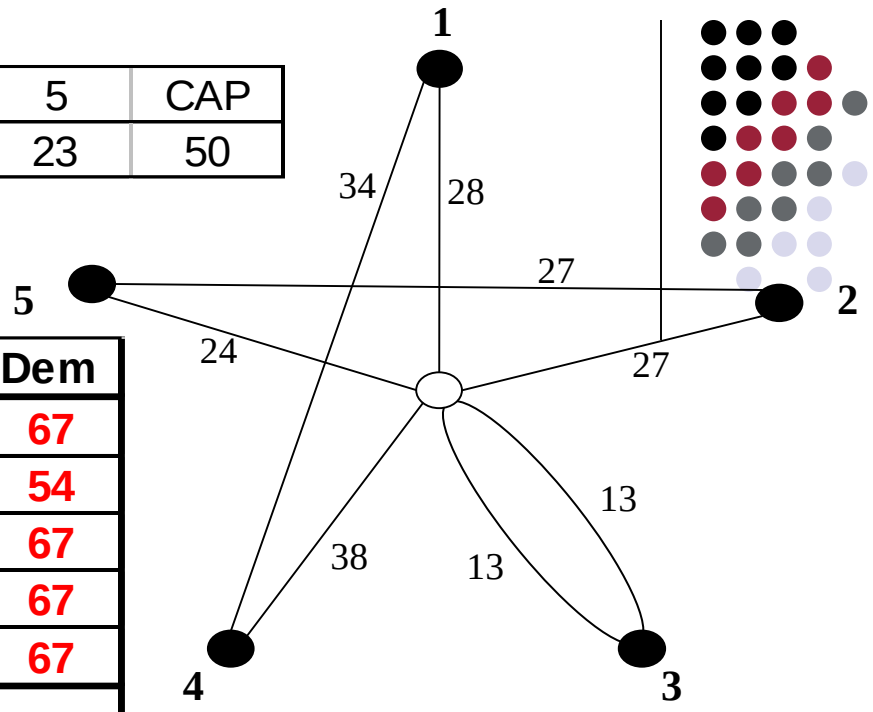
Cidades	1	2	3	4	5	CAP
Demanda	15	17	27	12	23	50



i	j	d_{i0}	d_{j0}	d_{ij}	S_{ij}	Dem
1	2	28	27	52	3	44
1	3	28	13	32	9	**54**
1	5	28	24	52	0	50
2	3	27	13	20	20	44
2	4	27	38	43	22	44
2	**5**	**27**	**24**	**27**	**24**	**40**
3	4	13	38	28	23	**54**
3	5	13	24	32	5	50
4	5	38	24	43	19	50

Total percorrido:	228
Nº de caminhões:	4

Cidades	1	2	3	4	5	CAP
Demanda	15	17	27	12	23	50



i	j	d_{i0}	d_{j0}	d_{ij}	S_{ij}	Dem
1	2	28	27	52	3	67
1	3	28	13	32	9	54
1	5	28	24	52	0	67
2	3	27	13	20	20	67
2	4	27	38	43	22	67
3	**4**	**13**	**38**	**28**	**23**	54
3	5	13	24	32	5	67
4	5	38	24	43	19	67

Total percorrido:	204
Nº de caminhões:	3

Bibliografia

- Souza, M. J. F. **Inteligência Computacional para Otmização**. Disponível em www.iceb.ufop.br/decom/prof/marcone/ico2009, acessado em Agosto de 2019.

- Martins, A. X. **Heurísticas e Metaheurísticas**. Material didático, 2018.