

# Inteligência Computacional para Otimização

Marcone Jamilson Freitas Souza  
Departamento de Computação  
Universidade Federal de Ouro Preto  
<http://www.iceb.ufop.br/prof/marcone>

# Classificação dos métodos heurísticos

## ✓ Construtivos

Constroem uma solução passo a passo, elemento por elemento

## ✓ de refinamento

Consistem em melhorar uma solução, através de modificações em seus elementos

# Heurística de construção gulosa (Heurística clássica)

## ◆ Funcionamento:

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

# Uma heurística construtiva gulosa para o Problema da Mochila

1º Passo: Calcular a relação benefício/peso

Pessoa	Peso (Kg)	Benefício	Benefício/ Peso
cruzeirense	140	0	0
Recém-graduado	60	1	0,017
ATLETICANO	100	3	0,030
Professor de geografia	80	4	0,050
Morena “olhos verdes”	75	3	0,040
Loira	60	2	0,030
Marcone	90	10	0,111

# Uma heurística construtiva gulosa para o Problema da Mochila

2º Passo: Ordenar os elementos

<b>Pessoa</b>	<b>Peso (Kg)</b>	<b>Benefício</b>	<b>Benefício/ Peso</b>
Marcone	90	10	0,111
Professor de geografia	80	4	0,050
Morena “olhos verdes”	75	3	0,040
Loira	60	2	0,030
ATLETICANO	100	3	0,030
Recém-graduado	60	1	0,017
cruzeirense	140	0	0

# Uma heurística construtiva gulosa para o Problema da Mochila

3º Passo: Escolher o elemento que produzir a maior relação benefício/peso, e que respeite a capacidade do barco

Pessoa	Peso (Kg)	Benefício	Benefício/ Peso
Marcone	90	10	0,111
Professor de geografia	80	4	0,050
Morena “olhos verdes”	75	3	0,040
Loira	60	2	0,030
ATLETICANO	100	3	0,030
Recém-graduado	60	1	0,017
cruzeirense	140	0	0

# Uma heurística construtiva gulosa para o Problema da Mochila

4º Passo: Repetir o passo anterior até que nenhum elemento possa ser colocado no barco sem ultrapassar a capacidade deste.

Pessoa	Peso (Kg)	Benefício	Benefício/ Peso
Marcone	90	10	0,111
Professor de geografia	80	4	0,050
Morena “olhos verdes”	75	3	0,040
Loira	60	2	0,030
ATLETICANO	100	3	0,030
Recém-graduado	60	1	0,017
cruzeirense	140	0	0

# Uma heurística construtiva gulosa para o Problema da Mochila

4º Passo: Repetir o passo anterior até que nenhum elemento possa ser colocado no barco sem ultrapassar a capacidade deste.

Pessoa	Peso (Kg)	Benefício	Benefício/ Peso
Marcone	90	10	0,111
Professor de geografia	80	4	0,050
Morena “olhos verdes”	75	3	0,040
Loira	60	2	0,030
ATLETICANO	100	3	0,030
Recém-graduado	60	1	0,017
cruzeirense	140	0	0



# Heurística de construção gulosa

**procedimento** *ConstrucaoGulosa*( $g(\cdot)$ ,  $s$ );

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*;

\* Considera-se um problema de maximização

# Exemplificando a aplicação da heurística construtiva gulosa

Seja, então, uma mochila de capacidade  $b = 23$  e os 5 objetos da tabela abaixo, com os respectivos pesos e benefícios.

Objeto ( $j$ )	1	2	3	4	5
Peso ( $w_j$ )	4	5	7	9	6
Benefício ( $p_j$ )	2	2	3	4	4

Construamos uma solução para esse problema usando a seguinte idéia: adicionemos à mochila a cada passo, o objeto mais valioso por unidade de peso e que não ultrapasse a capacidade da mochila. Reordenando os objetos de acordo com a relação  $p_j/w_j$ , obtemos:

Objeto ( $j$ )	5	1	4	3	2
Peso ( $w_j$ )	6	4	9	7	5
Benefício ( $p_j$ )	4	2	4	3	2
$(p_j/w_j)$	0.67	0.50	0.44	0.43	0.40

Representemos uma solução  $s$  por um vetor binário de  $n$  posições.

# Exemplificando a aplicação da heurística construtiva gulosa

**Passo 1** : Adicionemos, primeiramente, o objeto 5, que tem a maior relação  $p_j/w_j$

$$s = (00001)^t$$

$$f(s) = 4$$

$$\text{Peso corrente da mochila} = 6 < b = 23$$

**Passo 2** : Adicionemos, agora, o objeto 1, que tem a segunda maior relação  $p_j/w_j$

$$s = (10001)^t$$

$$f(s) = 6$$

$$\text{Peso corrente da mochila} = 10 < b = 23$$

**Passo 3** : Adicionemos, agora, o objeto 4, que tem a terceira maior relação  $p_j/w_j$

$$s = (10011)^t$$

$$f(s) = 10$$

$$\text{Peso corrente da mochila} = 19 < b = 23$$

**Passo 4** : O objeto a ser alocado agora seria o terceiro. No entanto, esta alocação faria superar a capacidade da mochila. Neste caso, devemos tentar alocar o próximo objeto com a maior relação  $p_j/w_j$ , que é o objeto 1. Como também a alocação desse objeto faria superar a capacidade da mochila e não há mais objetos candidatos, concluímos que a solução anterior é a solução final, isto é:  $s^* = (10011)^t$  com  $f(s^*) = 10$ .

# Um modelo de programação matemática para o Problema da Mochila

Sejam:

$n$  elementos

$c$  = capacidade da mochila

$b_i$  = benefício do elemento  $i$

$p_i$  = peso do elemento  $i$

Variável de decisão:

$x_i = \begin{cases} 1 & \text{se o elemento } i \text{ for colocado na mochila} \\ 0 & \text{caso contrário} \end{cases}$

# Um modelo de programação matemática para o Problema da Mochila

$$\max \sum_{i=1}^n b_i x_i$$

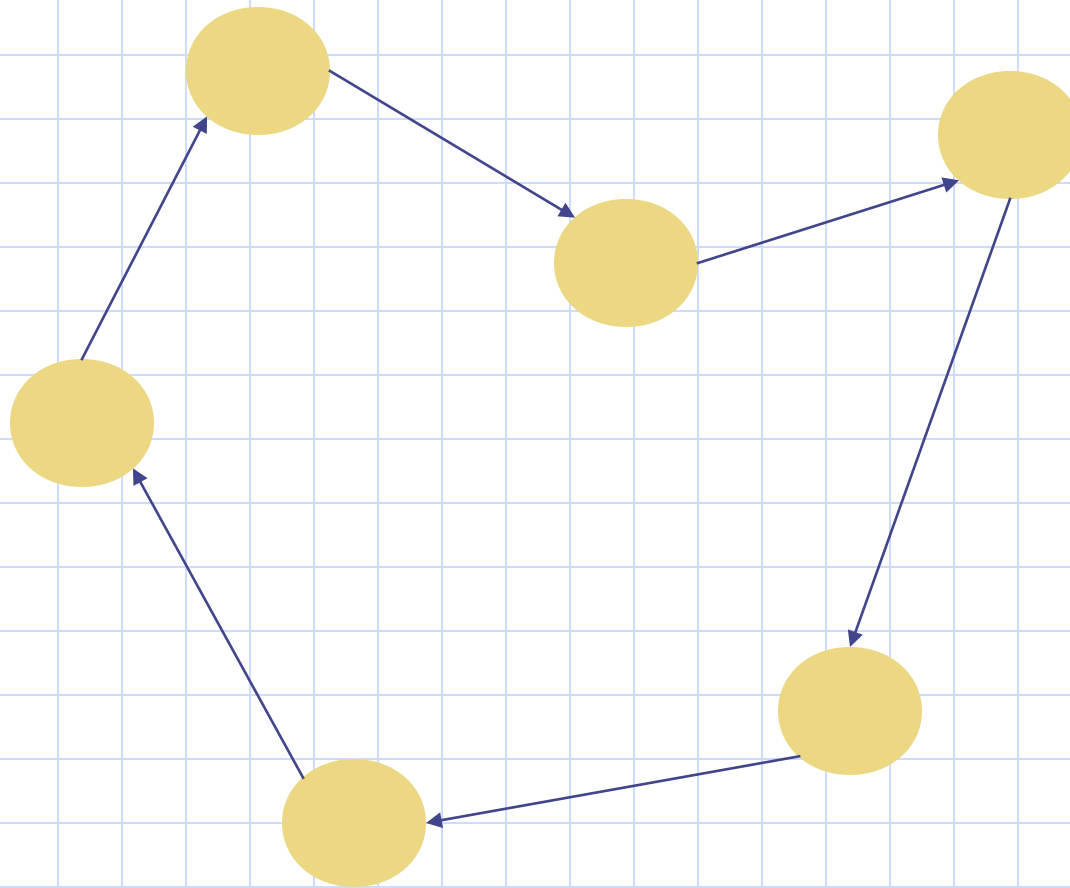
$$\sum_{i=1}^n p_i x_i \leq c$$

$$x_i \in \{0,1\} \quad \forall i=1,\dots,n$$

# Problema do Caixeiro Viajante

- ◆ Dado um conjunto de cidades e uma matriz de distâncias entre elas
- ◆ PCV consiste em encontrar uma rota para um Caixeiro Viajante tal que este:
  - parta de uma cidade origem
  - passe por todas as demais cidades uma única vez
  - retorne à cidade origem ao final do percurso
  - percorra a menor distância possível
- ◆ Rota conhecida como ciclo hamiltoniano

# Problema do Caixeiro Viajante



# Formulação Matemática para o Problema do Caixeiro Viajante

- Dados de entrada:
  - Cidades: Conjunto de cidades
  - $d_{ij}$  = distância entre as cidades  $i$  e  $j$
- 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:

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

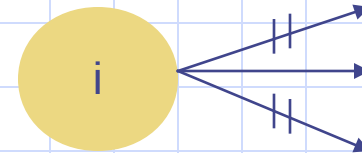


# Formulação Matemática para o Problema do Caixeiro Viajante

Restrições:

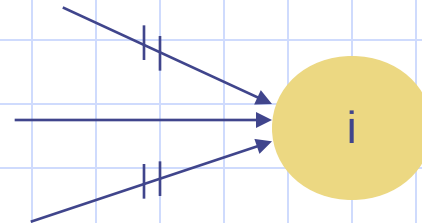
1. De cada cidade  $i$  só sai uma aresta:

$$\sum_{j \in \text{Cidades}} x_{ij} = 1 \quad \forall i \in \text{Cidades}$$



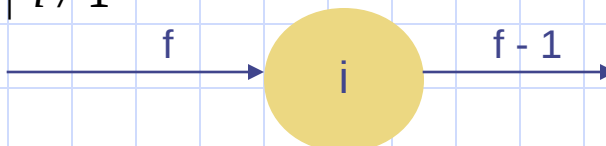
2. A cada cidade  $j$  só chega uma aresta:

$$\sum_{i \in \text{Cidades}} x_{ij} = 1 \quad \forall j \in \text{Cidades}$$



# Formulação Matemática para o Problema do Caixeiro Viajante

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

$$\sum_{j \in \text{Cidades}} f_{ji} - \sum_{j \in \text{Cidades}} f_{ij} = 1 \quad \forall i \in \text{Cidades} \mid i \neq 1$$


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

$$f_{ij} \leq (n-1) x_{ij} \quad \forall i \in \text{Cidades}, \forall j \in \text{Cidades}$$

5. Integralidade e não negatividade:

$$f_{ij} \geq 0 \quad \forall i \in \text{Cidades}, \forall j \in \text{Cidades}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in \text{Cidades}, \forall j \in \text{Cidades}$$

# Problema do Caixeiro Viajante: Complexidade

- ◆ Considerando PCV simétrico ( $d_{ij} = d_{ji}$ ), para  $n$  cidades há  $(n-1)!/2$  rotas possíveis
- ◆ Para  $n = 20$  cidades, há  $10^{16}$  rotas possíveis. Um computador que avalia uma rota em  $10^{-8}$  segundos gastaria cerca de 19 anos para encontrar a melhor solução por enumeração completa
- ◆ Métodos de enumeração implícita, tais como *branch-and-bound*, embutidos em *solvers*, podem reduzir significativamente este tempo
- ◆ Não há garantia de que isso sempre ocorra

# Problema do Caixeiro Viajante Complexidade

- ◆ Para dimensões mais elevadas, a resolução por métodos de programação matemática é proibitiva em termos de tempos computacionais
- ◆ PCV é da classe NP-difícil: não existem algoritmos exatos que o resolvam em tempo polinomial
- ◆ À medida que  $n$  cresce, o tempo cresce exponencialmente
- ◆ PCV é resolvido por meio de heurísticas:
  - Procedimentos que seguem uma intuição para resolver o problema (forma humana de resolver o problema, fenômenos naturais, processos biológicos, etc.)
  - Não garantem a otimalidade da solução final
  - Em geral, produzem soluções finais de boa qualidade rapidamente

# Heurísticas construtivas para o Problema do Caixeiro Viajante

## ◆ Vizinho mais próximo

- Idéia 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

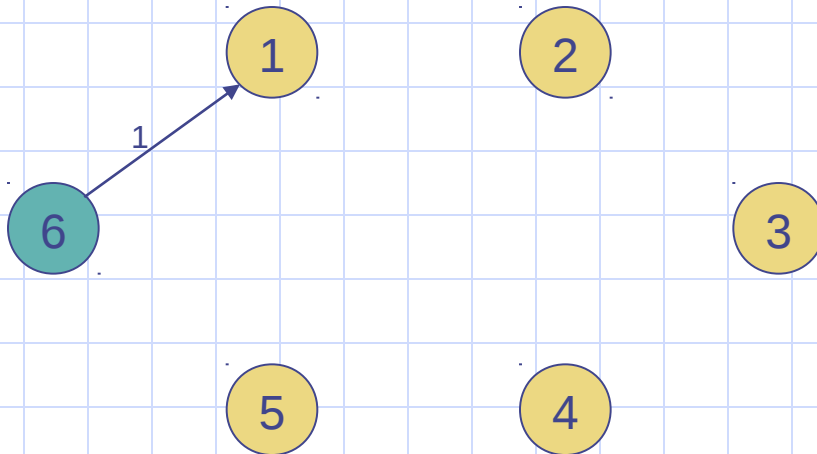
- Idéia central: Construir uma rota passo a passo, partindo de rota inicial envolvendo 3 cidades e adicionar a cada passo, a cidade  $k$  (ainda não visitada) entre a ligação  $(i, j)$  de cidades já visitadas, cujo custo de inserção seja o mais barato

# PCV – Vizinho mais Próximo

## Exemplo - Passo 1

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



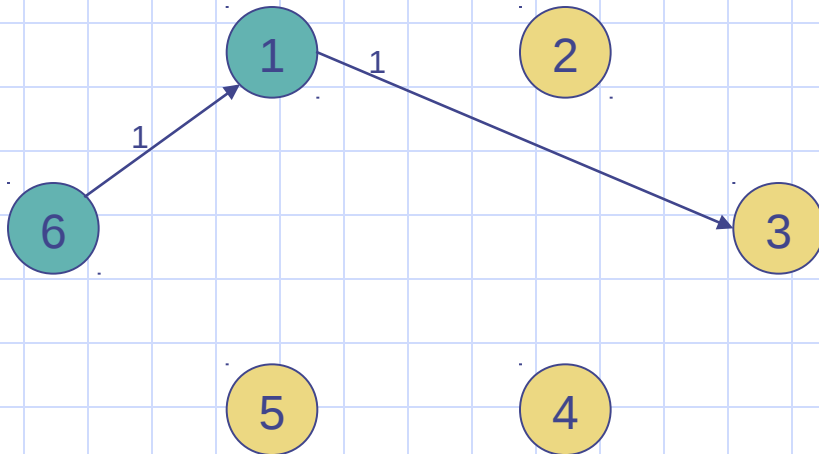
◆ Distância Total = 1

# PCV – Vizinho mais Próximo

## Exemplo - Passo 2

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



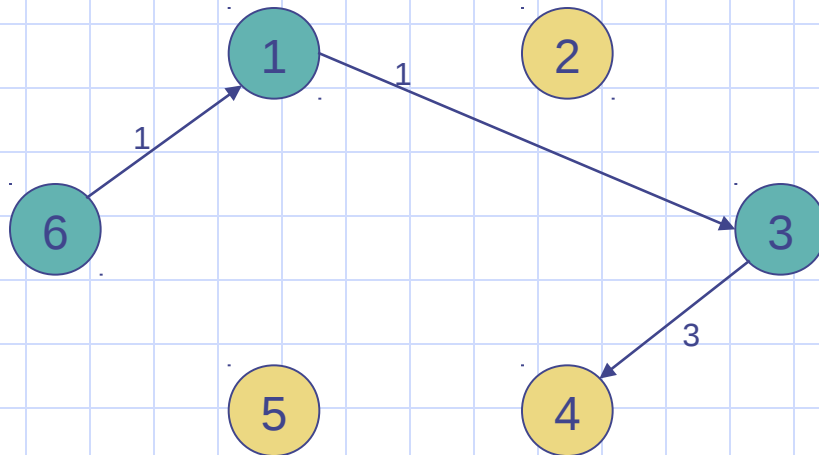
◆ **Distância Total** =  $1 + 1 = 2$

# PCV – Vizinho mais Próximo

## Exemplo - Passo 3

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

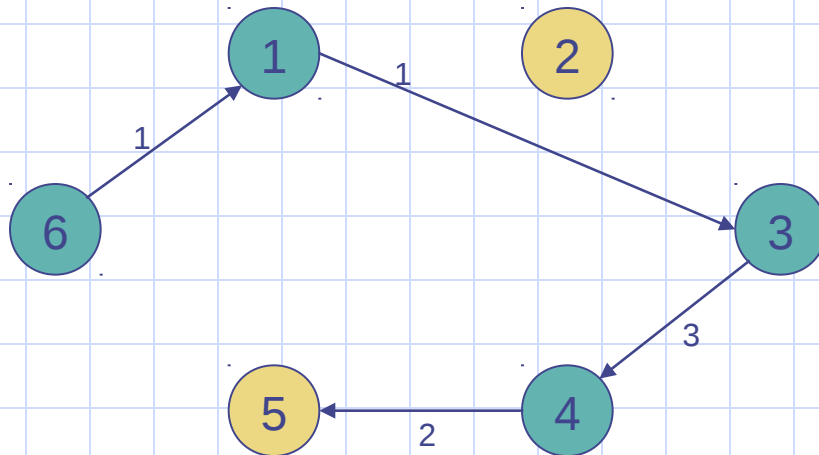


# PCV – Vizinho mais Próximo

## Exemplo - Passo 4

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



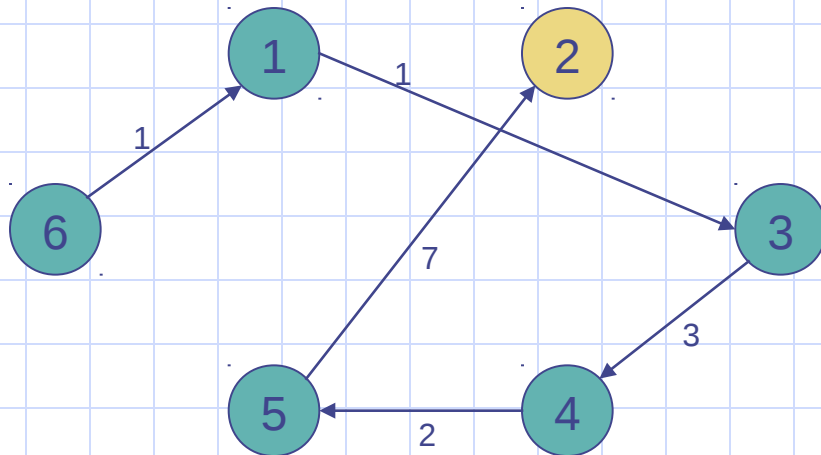
◆ **Distância Total** = 5 + 2 = 7

# PCV – Vizinho mais Próximo

## Exemplo - Passo 5

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

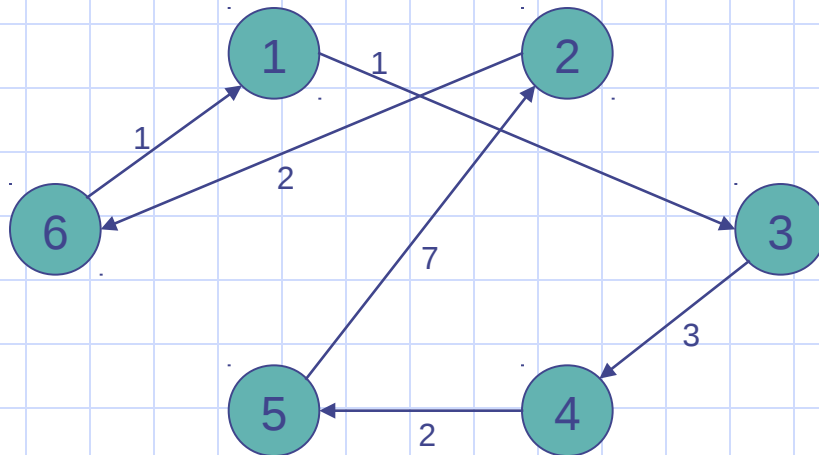


◆ **Distância Total = 7 + 7 = 14**

# PCV – Vizinho mais Próximo

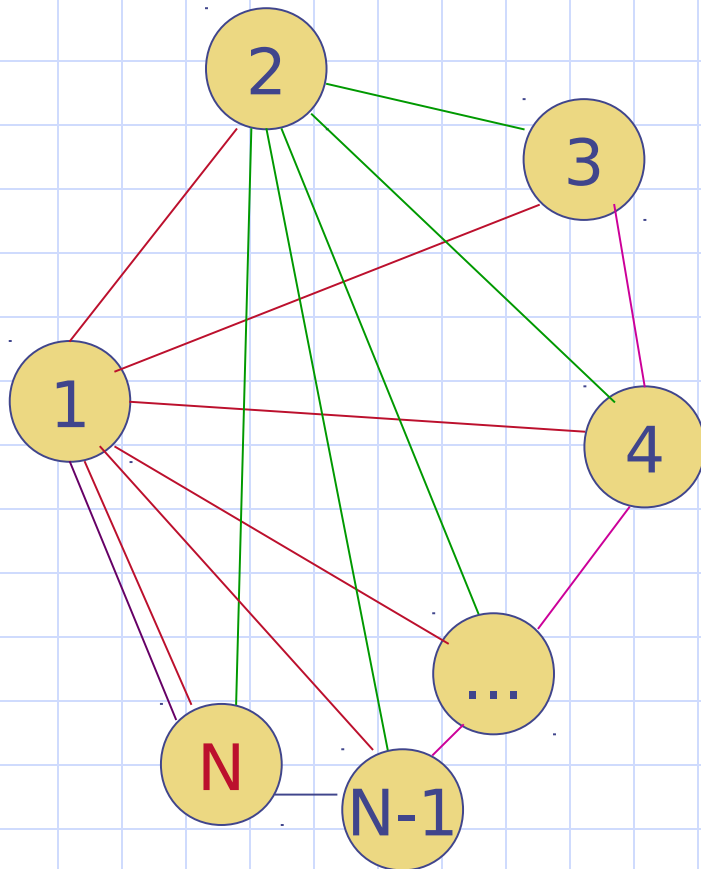
## Exemplo – Passo final: “Inserção forçada”

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



◆ **Distância Total** = 14 + 2 = 16

# Complexidade da heurística construtiva do vizinho mais próximo aplicada ao PCV



Iteração	Número de avaliações
1	$N-1$
2	$N-2$
...	...
$N-1$	1
$N$	1
Total	$1 + N(N-1)/2$

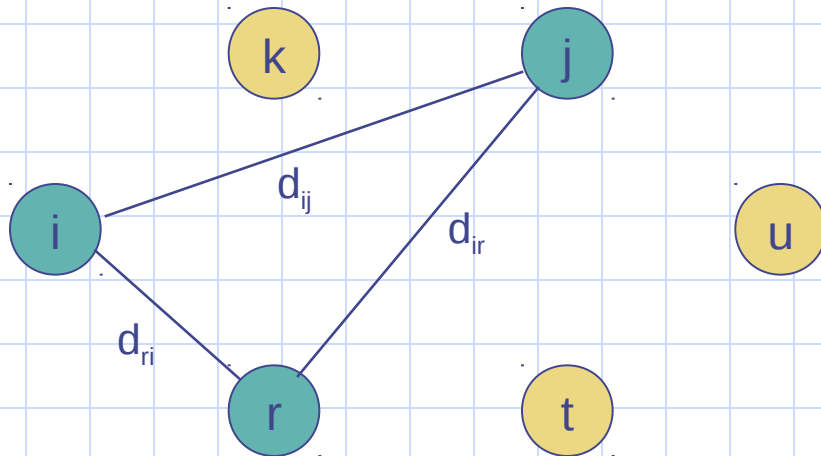
# Heurística da Inserção Mais Barata para o Problema do Caixeiro Viajante

## ◆ Inserção mais barata

- Idéia 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

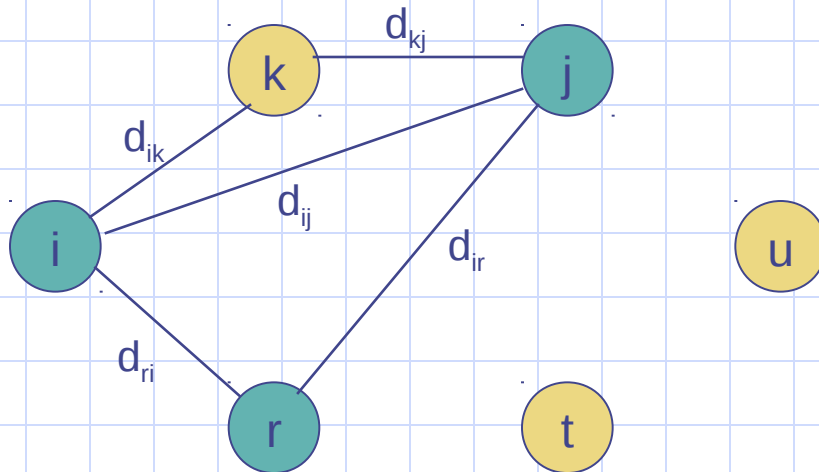
# Heurística da Inserção Mais Barata para o Problema do Caixeiro Viajante

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



# Heurística da Inserção Mais Barata para o Problema do Caixeiro Viajante

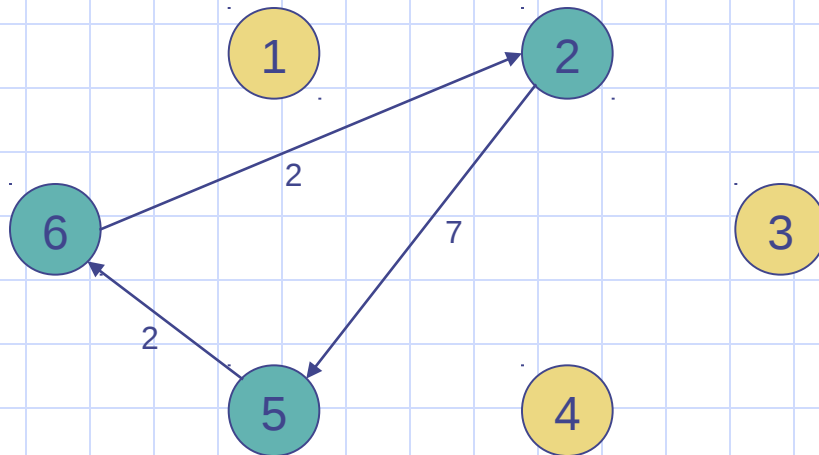
Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0



# PCV – Inserção mais Barata

## Exemplo - Passo 1

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



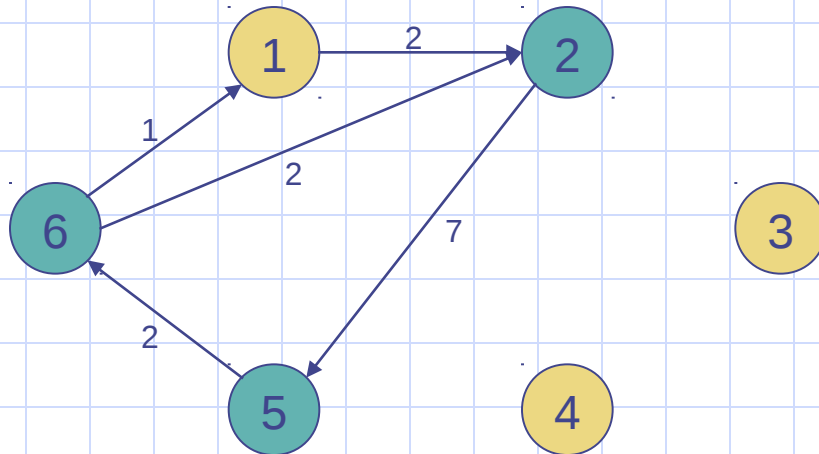
◆ **Distância Total = 11**



# PCV – Inserção mais Barata

## Exemplo - Passo 2

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



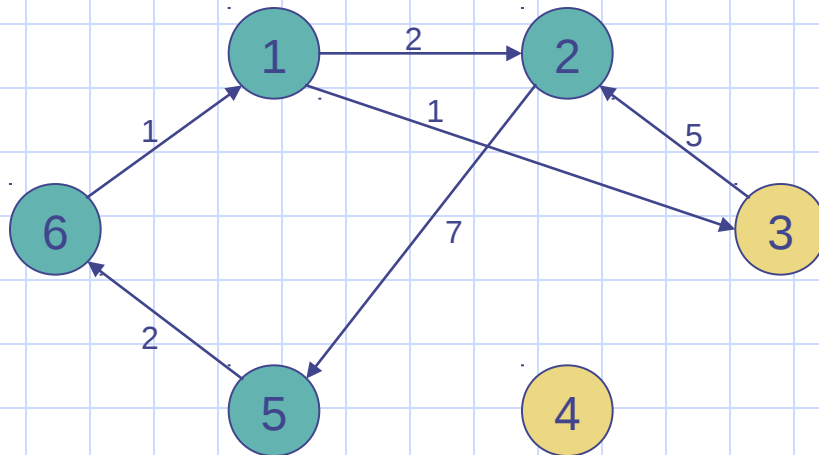
i	k	j	$d_{ik} + d_{kj} - d_{ij}$
6	1	2	$1 + 2 - 2 = 1$
6	3	2	$6 + 5 - 2 = 9$
6	4	2	$6 + 9 - 2 = 3$
2	1	5	$2 + 9 - 7 = 4$
2	3	5	$5 + 8 - 7 = 6$
2	4	5	$9 + 2 - 7 = 4$
5	1	6	$9 + 1 - 2 = 8$
5	3	6	$8 + 6 - 2 = 12$
5	4	6	$2 + 6 - 2 = 6$

◆ **Distância Total** = 11 + 1 = 12

# PCV – Inserção mais Barata

## Exemplo - Passo 3

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	$2 + 6 - 2 = 6$

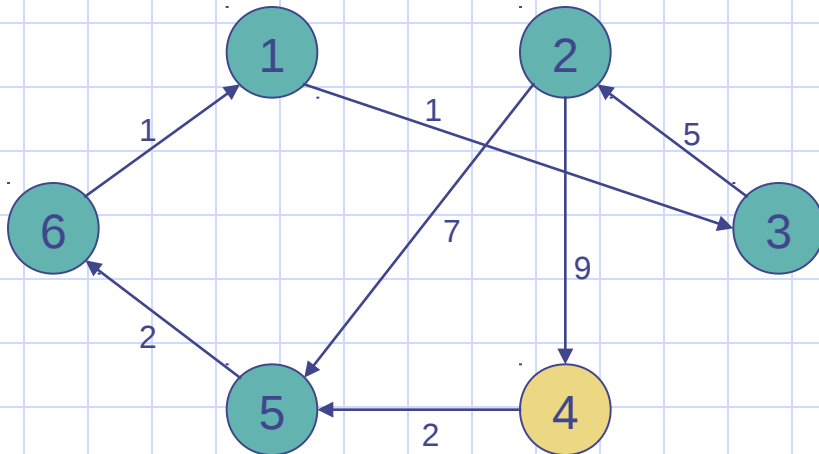
◆ **Distância Total** =  $12 + 4 = 16$

# PCV – Inserção mais Barata

## Exemplo – Passo final

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	$2 + 6 - 2 = 6$

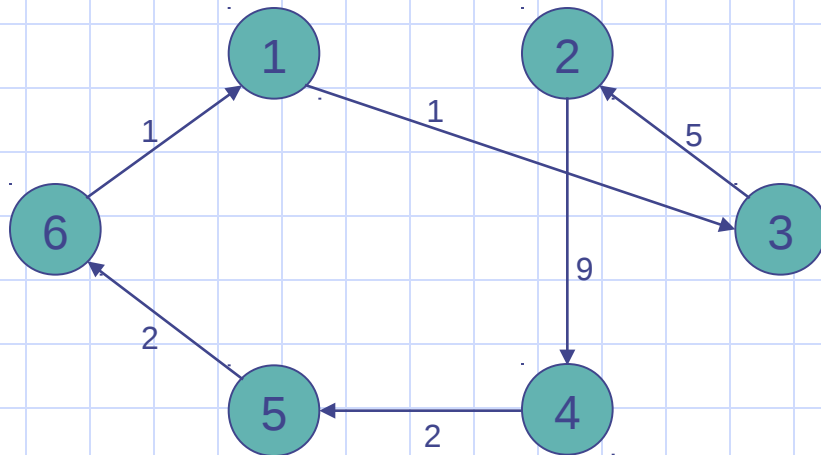


◆ **Distância Total** =  $16 + 4 = 20$

# PCV – Inserção mais Barata

## Exemplo – Solução final

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



◆ **Distância Total** = 16 + 4 = 20

$s = (6 \ 1 \ 3 \ 2 \ 4 \ 5)$

# Complexidade da heurística construtiva da inserção mais barata aplicada ao PCV

Iteração	Número de avaliações
1	$3(N - 3)$
2	$4(N - 4)$
...	...
$i-2$	$i(N-i)$
...	...
$N-3$	$(N - 1)(N-(N-1))$
Total	$\sum_{i=3}^{n-1} i(n-i)$

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

# Comparação entre as heurísticas construtivas para o PCV

Exemplo para  $n = 20$  cidades

Método	Complexidade	Tempo (s)
Vizinho mais próximo	$\frac{1}{2}n^2 - n + 1$	$1,8 \times 10^{-6}$
Inserção mais barata	$\frac{1}{6}n^3 - n^2 - \frac{5}{6}n - 3$	$9,1 \times 10^{-6}$

Supor uma avaliação executada em  $10^{-8}$  segundos

# Comparação entre as heurísticas construtivas para o PCV

Exemplo para  $n = 1000$  cidades

Método	Complexidade	Tempo (s)
Vizinho mais próximo	$\frac{1}{2}n^2 - n + 1$	0,005
Inserção mais barata	$\frac{1}{6}n^3 - n^2 - \frac{5}{6}n - 3$	1,665

Supor uma avaliação executada em  $10^{-8}$  segundos

# Comparação entre as heurísticas construtivas para o PCV

Exemplo para  $n = 10000$  cidades

Método	Complexidade	Tempo (s)
Vizinho mais próximo	$\frac{1}{2}n^2 - n + 1$	0,5
Inserção mais barata	$\frac{1}{6}n^3 - n^2 - \frac{5}{6}n - 3$	1665

Supor uma avaliação executada em  $10^{-8}$  segundos



# Problema de Roteamento de Veículos

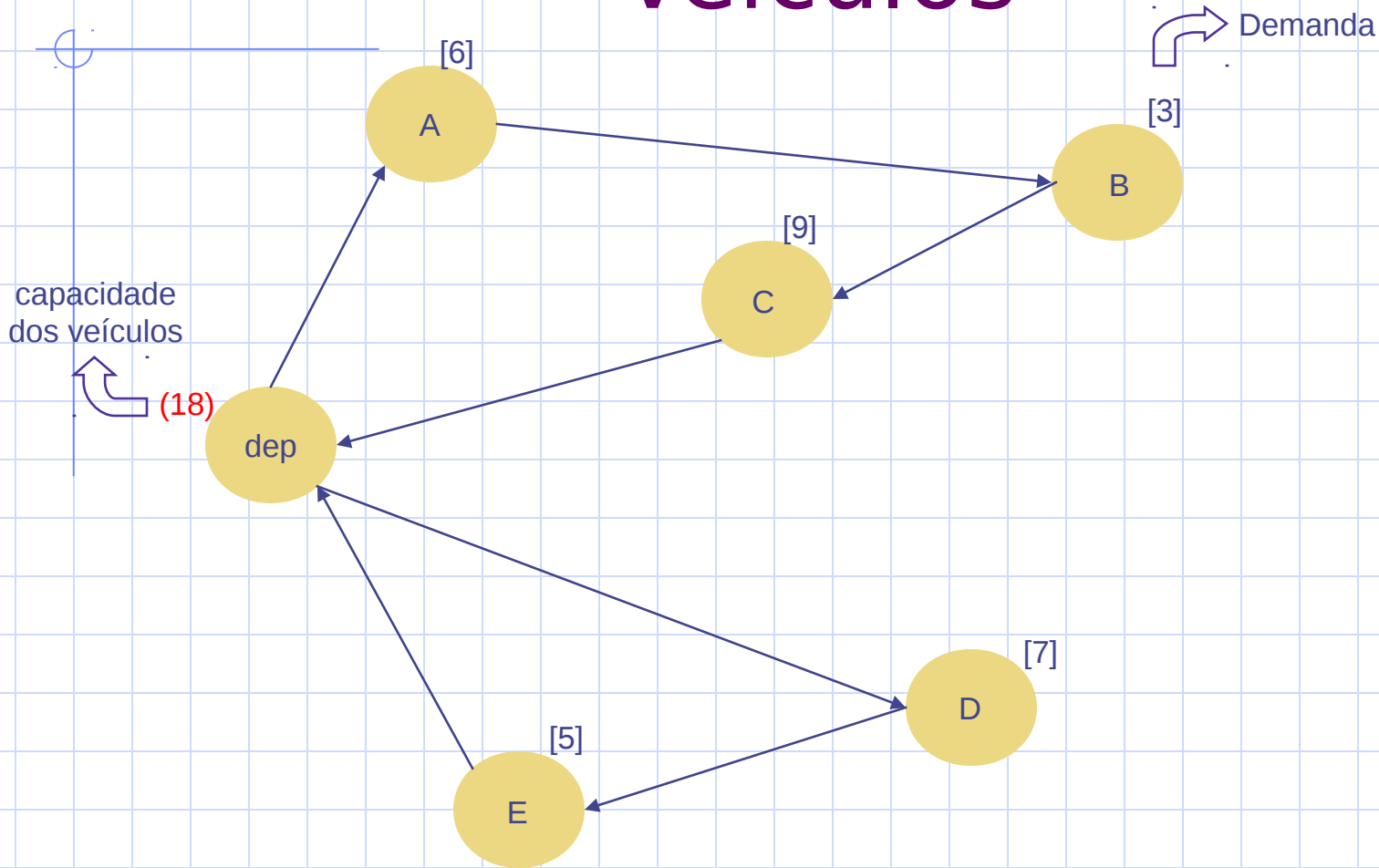
## ◆ Dados de entrada:

- Um depósito
- Uma frota de veículos, com base no depósito
- Um conjunto de clientes
- A demanda dos clientes
- Uma matriz de distâncias  $D = (d_{ij})$  entre depósito e clientes e entre pares de clientes
- Cidades = depósito  $\cup$  Clientes

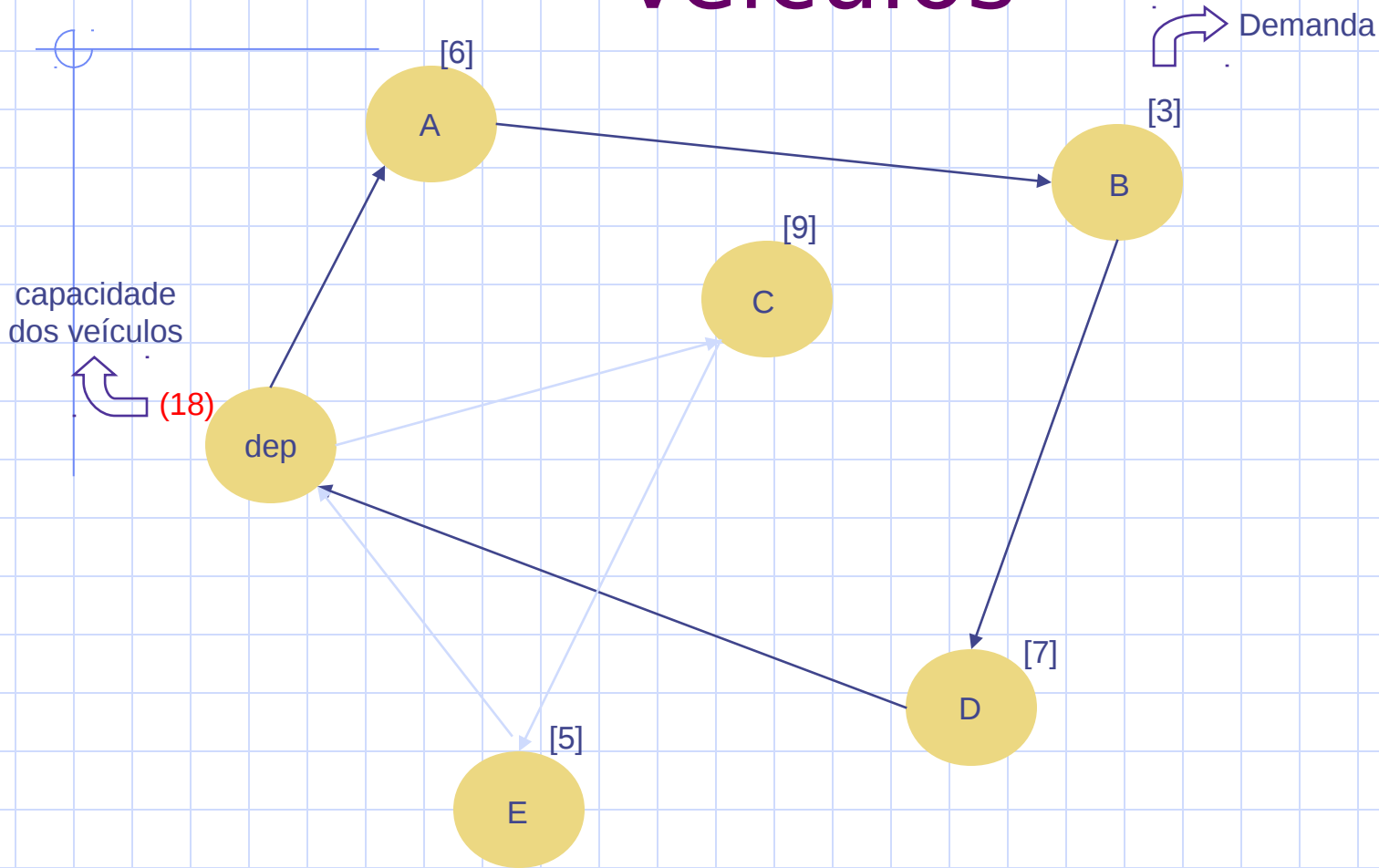
## ◆ PRV consiste em encontrar um conjunto de rotas para os veículos tal que:

- Cada rota comece e termine no depósito
- Cada cliente seja atendido por um único veículo
- A capacidade ( $cap_{Veic}$ ) dos veículos seja respeitada
- A distância total percorrida seja a menor possível

# Problema de Roteamento de Veículos



# Problema de Roteamento de Veículos



# Formulação Matemática para o Problema de Roteamento de Veículos

- Dados de entrada:
  - Cidades: Conjunto formado por Depósito e Clientes
  - $d_{ij}$  = distância entre as cidades  $i$  e  $j$
  - $\text{demanda}_i$  = demanda da cidade  $i$  ( $\text{demanda}_{\text{dep}} = 0$ )
- 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:

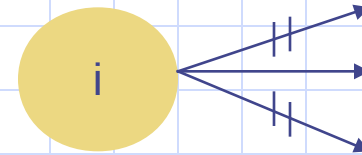
$$\min \sum_{i \in \text{Cidades}} \sum_{j \in \text{Cidades}} d_{ij} x_{ij}$$

# Formulação Matemática para o Problema de Roteamento de Veículos

Restrições:

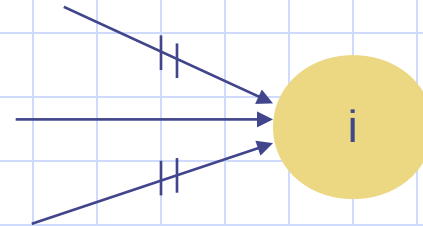
1. De cada cidade  $i$ , exceto o depósito (1), só sai um veículo:

$$\sum_{j \in \text{Cidades}} x_{ij} = 1 \quad \forall i \in \text{Cidades}, i \neq 1$$



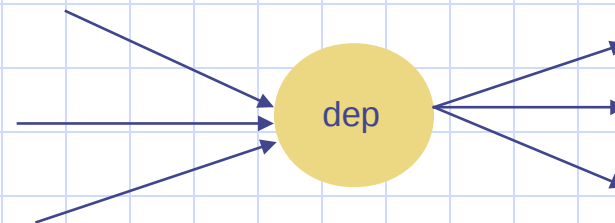
2. A cada cidade  $j$ , exceto o depósito (1), só chega um veículo:

$$\sum_{i \in \text{Cidades}} x_{ij} = 1 \quad \forall j \in \text{Cidades}, j \neq 1$$



3. O número de veículos que saem do depósito é igual ao que chegam ao depósito:

$$\sum_{j \in \text{Cidades}} x_{1j} = \sum_{j \in \text{Cidades}} x_{j1}$$



# Formulação Matemática para o Problema de Roteamento de Veículos

4. Ao passar por uma cidade  $j$ , exceto o depósito, o veículo deve atender a demanda dessa cidade, isto é, deve deixar  $demandaj$  unidades de produto na cidade  $j$ ;

$$\sum_{i \in Cidades} f_{ij} - \sum_{i \in Cidades} f_{ji} = demandaj \quad \forall j \in Cidades \mid j \neq 1$$



5. O fluxo máximo em cada aresta não pode superar a capacidade do veículo:

$$f_{ij} \leq (capVeic) x_{ij} \quad \forall i \in Cidades, \forall j \in Cidades$$

6. Integralidade e não negatividade:

$$f_{ij} \geq 0 \quad \forall i \in Cidades, \forall j \in Cidades$$

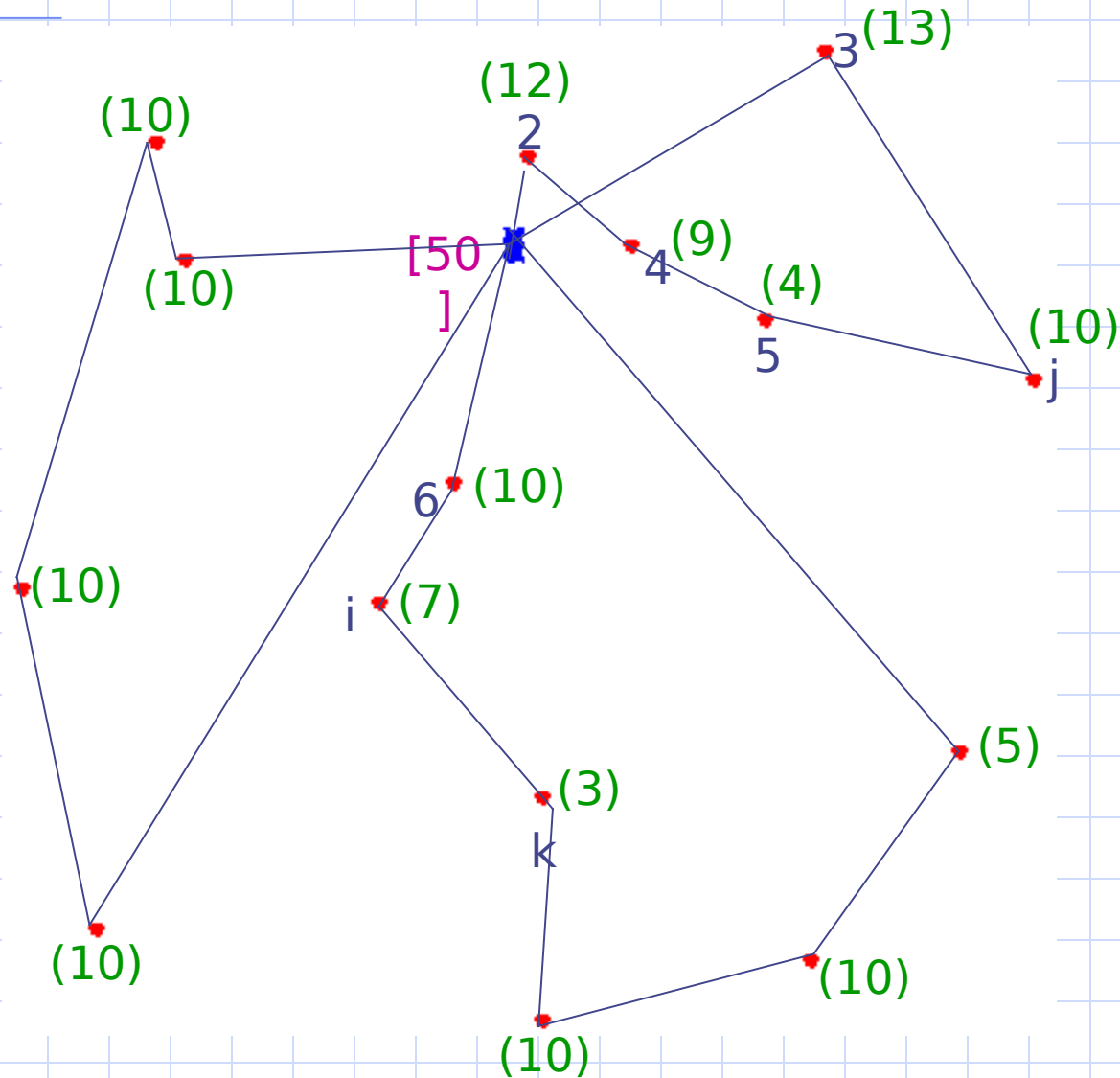
$$x_{ij} \in \{0,1\} \quad \forall i \in Cidades, \forall j \in Cidades$$

# Adaptação da Heurística do Vizinho mais próximo para o Problema de Roteamento de Veículos com frota homogênea

## ◆ Idéia básica:

- **Passo 1:** Partir do 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 do Vizinho mais Próximo Aplicada ao PRV





# Heurística Construtiva de Clark & Wright para o Problema de Roteamento de Veículos com frota homogênea

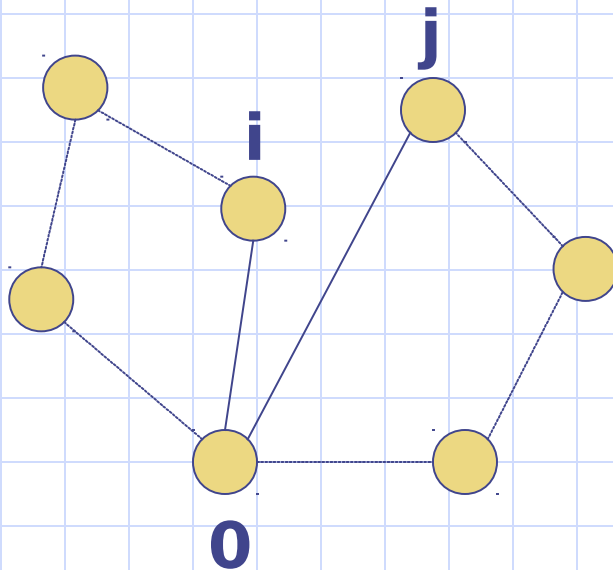
## ◆ Idéia básica:

- 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

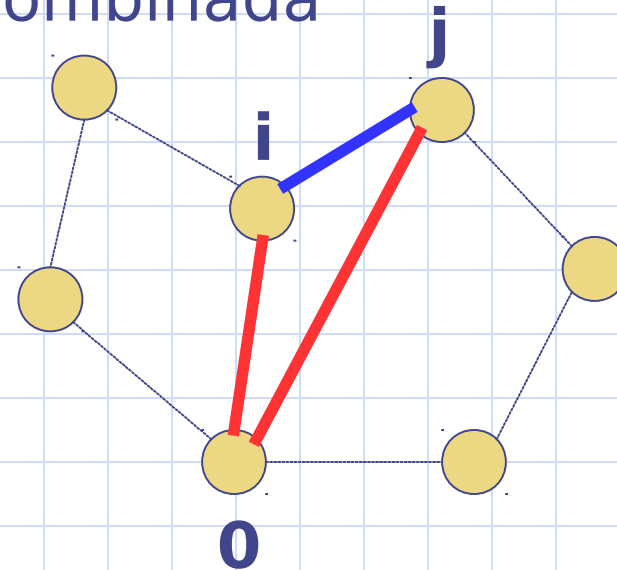
◆ À medida que se reduz a distância total percorrida, o número de veículos necessários também é reduzido

# Heurística Construtiva de Clark & Wright para o Problema de Roteamento de Veículos com frota homogênea

(a) Rota inicial



(b) Rota combinada

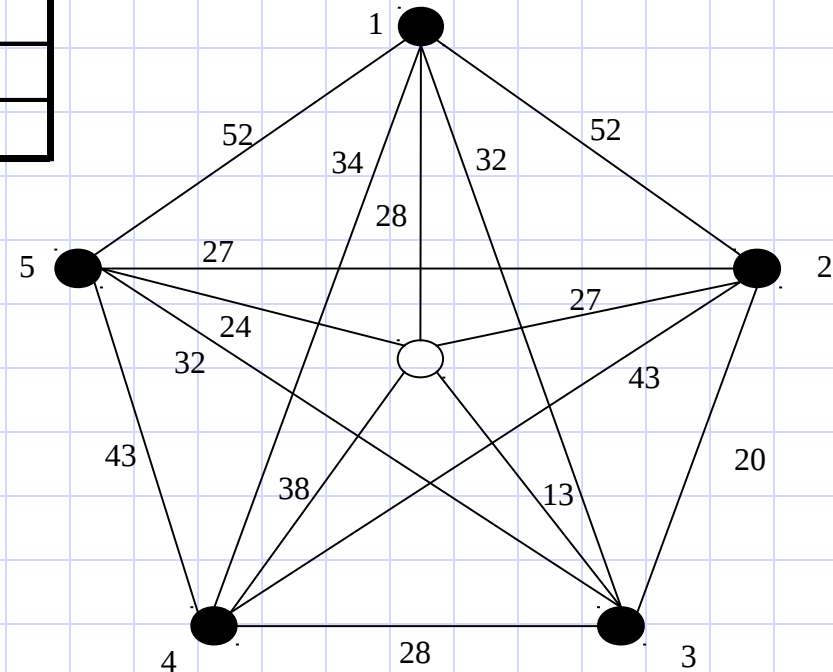


Economia (saving)  $s_{ij} = d_{i0} + d_{0j} - d_{ij}$

**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	$d_{i0}$	$d_{j0}$	$d_{ij}$	$s_{ij}$	Dem

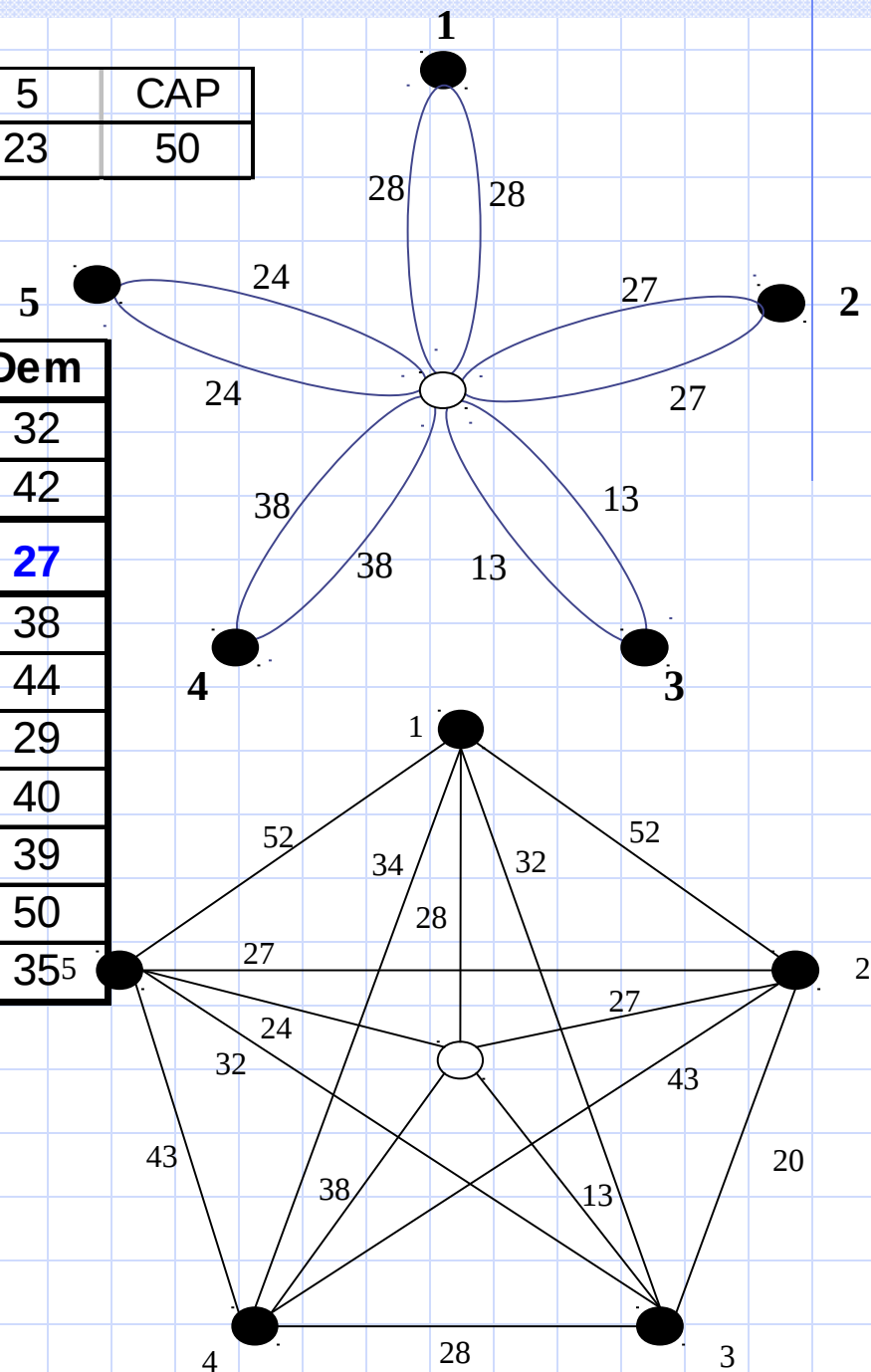


$$s_{ij} = d_{i0} + d_{j0} - d_{ij}$$

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

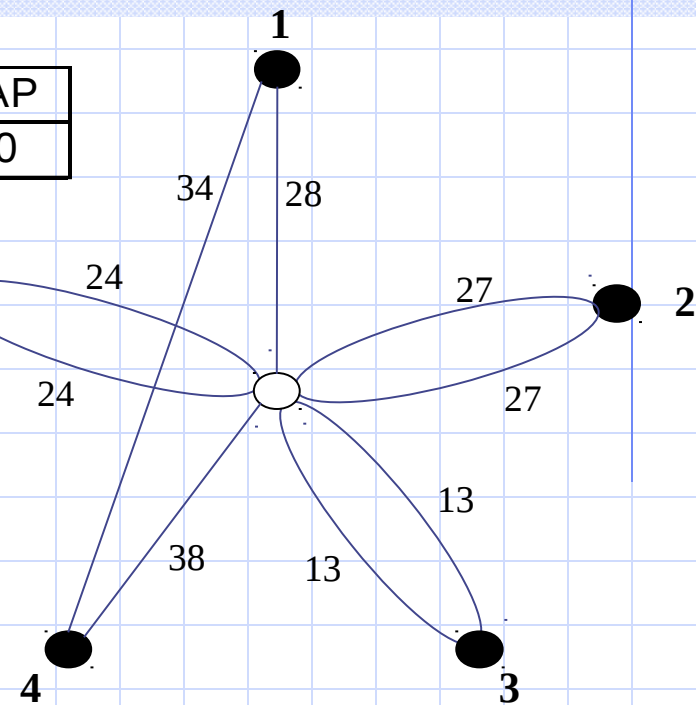
i	j	$d_{i0}$	$d_{j0}$	$d_{ij}$	$s_{ij}$	Dem
1	2	28	27	52	3	32
1	3	28	13	32	9	42
<b>1</b>	<b>4</b>	<b>28</b>	<b>38</b>	<b>34</b>	<b>32</b>	<b>27</b>
1	5	28	24	52	0	38
2	3	27	13	20	20	44
2	4	27	38	43	22	29
2	5	27	24	27	24	40
3	4	13	38	28	23	39
3	5	13	24	32	5	50
4	5	38	24	43	19	35

Total percorrido:	260
Nº de caminhões:	5



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

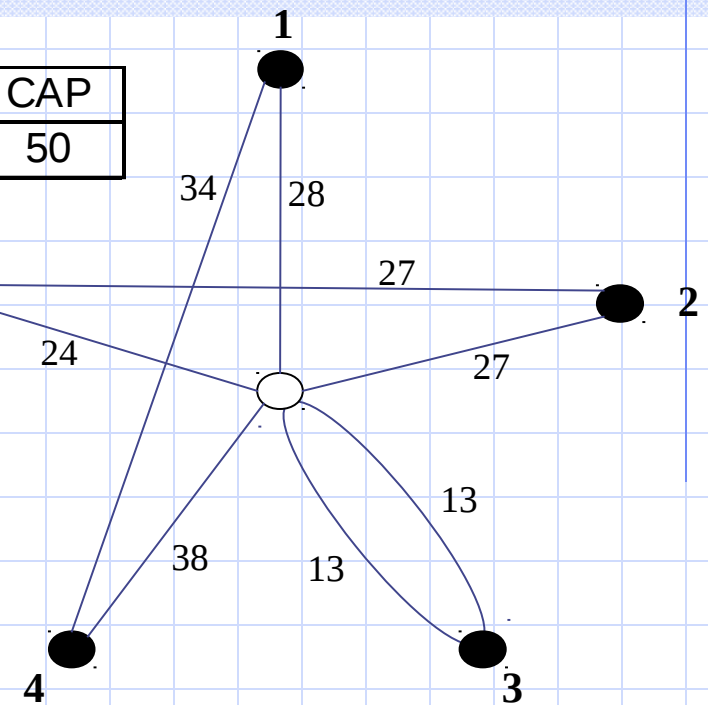
i	j	$d_{i0}$	$d_{j0}$	$d_{ij}$	$s_{ij}$	Dem
1	2	28	27	52	3	44
1	3	28	13	32	9	54
1	5	28	24	52	0	50
2	3	27	13	20	20	44
2	4	27	38	43	22	44
2	5	27	24	27	24	40
3	4	13	38	28	23	54
3	5	13	24	32	5	50
4	5	38	24	43	19	50



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