Heurísticas

Fonte: aulas dos profs. Alysson Costa, Vitória Pureza, Vinícius Armentano e Maristela Santos

Métodos exatos

- Objetivo: encontrar a solução ótima para um dado problema.

- Por exemplo:
  - Branch-and-bound
  - Plano de Cortes
  - Outros …

Heurística

O livro clássico, "How to Solve It" de Polya (1957), com versão em português, é considerado como o texto de maior influência no re-descobrimento de heurísticas, ou estudo de métodos de descoberta e invenção. Neste livro, o autor já comenta sobre o "intelligent problem-solver".

* Armentano, V.A. - Slide

Heurísticas

Nicholson (1971, ver artigo de Osman). Procedimento para resolver problemas através de um enfoque "intuitivo", em geral racional, no qual a estrutura do problema possa ser interpretada e explorada inteligentemente para obter uma solução razoável.

* Armentano, V.A. - Slide

Heurísticas

Heurísticas são estudadas hoje em dia por duas áreas: Inteligência Artificial (IA) e Pesquisa Operacional (PO).

Diferenças de Enfoques entre IA e PO

IA	PO
ciência da computação	matemática
mais empírico	mais teórico
métodos de busca	métodos especializados
representação com grafos	representação com modelos matemáticos

* Armentano, V.A. - Slide

Métodos heurísticos

- Objetivo: encontrar uma solução de **boa** qualidade para um dado problema em tempo computacional **razoável.**

- São desenvolvidos especificamente para um dado problema.

Heurísticas

- Muitas vezes são baseadas em procedimentos simples, fáceis de implementar e fáceis de "justificar".

- Muito aceitas em contextos práticos. (Métodos implementados em pacotes comerciais – e.g., para resolução de problemas de roteamento de veículos – muitas vezes obtém soluções a 10, 20% das melhores soluções possíveis).

Situações em que Heurísticas são apropriadas

a) Quando métodos exatos são proibitivos: tempo de execução, memória, tempo de desenvolvimento.

b) Quando os dados têm pouca confiabilidade: uma solução "ótima" não tem sentido.

c) Quando o modelo matemático não representa aspectos importantes do problema real; em um caso extremo de um problema muito complexo, não é possível uma representação por modelo matemático. Neste sentido, heurísticas são muito flexíveis.

* Armentano, V.A. - Slide

Situações em que Heurísticas são apropriadas

d) Para achar boas soluções de partida para métodos exatos: regra do canto noroeste para problemas de transporte formulados por programação linear, e soluções para métodos branch-and-bound ou branch-and-cut para programação inteira.

e) Para gerar soluções durante a resolução de métodos exatos.

f) Quando o decisor precisa compreender e interagir com o método para tomada de decisões.

* Armentano, V.A. - Slide

Heurísticas

VANTAGENS QUE JUSTIFICAM A UTILIZAÇÃO DE ALGORITMOS HEURÍSTICOS

- Utilizam uma fração do esforço computacional de um método exato.
- Permitem maior flexibilidade no manejo de características do problema.
- Geralmente oferecem mais de uma solução, permitindo ampliar as possibilidades de decisão, sobretudo quando existem fatores não quantificáveis que não podem ser incorporados no modelo, mas que precisam ser considerados.

Slide baseado em material da professora Vitória Pureza - UFSCAR

Heurísticas

DESVANTAGENS DE ALGORITMOS HEURÍSTICOS

☹ Em geral, **não é possível conhecer a qualidade da solução apresentada**, ou seja, quão próxima x_{heur} está da solução ótima x*. Em um problema de maximização, pode-se afirmar somente que $f(x_{heur}) \leq f(x^*)$.

☺ Existem métodos que podem fornecer **indicações sobre a qualidade das soluções da heurística** ⇒ ***Relaxação*** **do problema**, eliminando algumas das restrições, de forma que o problema seja mais fácil de resolver.

Relaxação linear - resolve o problema relaxado, ignorando as restrições de integralidade das variáveis. Se o ótimo do problema relaxado é $\mathbf{x^*_{relax}}$, então:

Ex.:

$$\mathbf{f(x_{heur}) \leq f(x^*) \leq f(x^*_{relax})} \text{ (maximização)}$$

Assim, **f(x*) próximos a $\mathbf{f(x_{heur})}$** garantem que a heurística está nos dando uma boa aproximação.

Slide baseado em material da professora Vitória Pureza - UFSCAR

Heurísticas

☺ Existem métodos que simplesmente **detectam que a heurística não é boa** ⇒ Uma possibilidade é a de gerar aleatoriamente várias soluções e verificar se têm qualidade similar à da solução heurística. Neste caso, põe-se em dúvida a efetividade da heurística.

Slide baseado em material da professora Vitória Pureza - UFSCAR

Características de boas heurísticas

- Qualidade: fornece soluções de alta qualidade com valor próximo ao valor ótimo.
- Robustez: heurística fornece resultados de alta qualidade para todas (ou a maioria) das instâncias de um problema.
- Rapidez: note o tradeoff qualidade $\times$ rapidez

* Armentano, V.A. - Slide

Características de boas heurísticas

- Simplicidade de implementação: note o tradeoff complexidade $\times$ qualidade $\times$ rapidez
- Simplicidade de compreensão
- Flexibilidade: pode ser adaptada facilmente para resolver outros problemas da mesma classe, como por exemplo, problemas de roteamento de veículos.

* Armentano, V.A. - Slide

Características de boas heurísticas

Journal of the Operational Research Society (2002) 53, 512–522

www.palgrave-journals.com/jors

A guide to vehicle routing heuristics

J-F Cordeau,[1] M Gendreau,[2] G Laporte,[1]* J-Y Potvin[2] and F Semet[3]

[1]*École des Hautes Études Commerciales, Montréal, Canada;* [2]*Université de Montréal, Montréal, Canada; and* [3]*Université de Valenciennes et du Hainaut Cambresis, Valenciennes, France*

Several of the most important classical and modern heuristics for the vehicle routing problem are summarized and compared using four criteria: accuracy, speed, simplicity and flexibility. Computational results are reported.
Journal of the Operational Research Society (2002) **53**, 512–522. DOI: 10.1057/palgrave/jors/2601319

Keywords: vehicle routing problem; heuristics

Métodos heurísticos

- Heurísticas construtivas
- Busca Local
- Etc.

Metaheurísticas

- GA
- GRASP
- TABU .... etc

Métodos heurísticos

- ✓ Construtivos
  - Constroem uma solução passo a passo, elemento por elemento
- ✓ de refinamento (melhoria)
  - 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

Heurísticas construtivas

Problema da mochila

a) Escolha dentre os itens ainda não selecionados, aquele com o maior valor por unidade de peso (v_i/p_i).

b) Repita (a) até que não caibam mais objetos.

Slide baseado em material da professora Vitória Pureza - UFSCAR

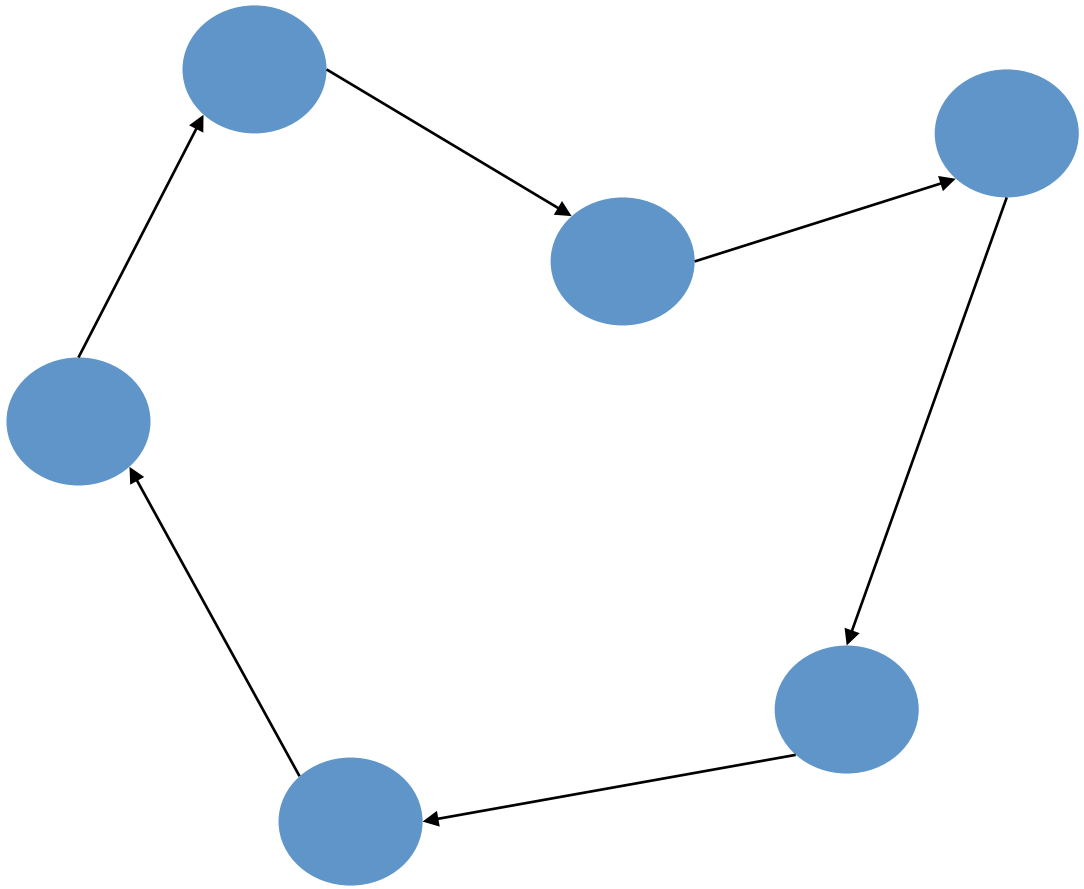
Problema do Caixeiro Viajante

- 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



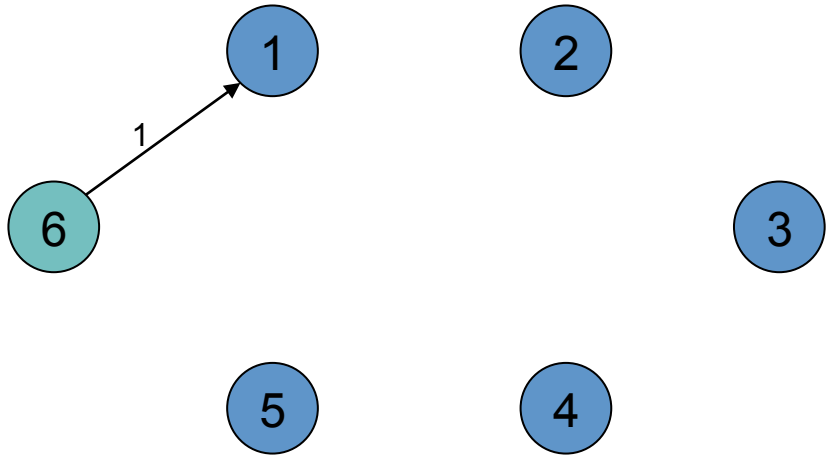
Heurísticas construtivas para o Problema do Caixeiro Viajante

- 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

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

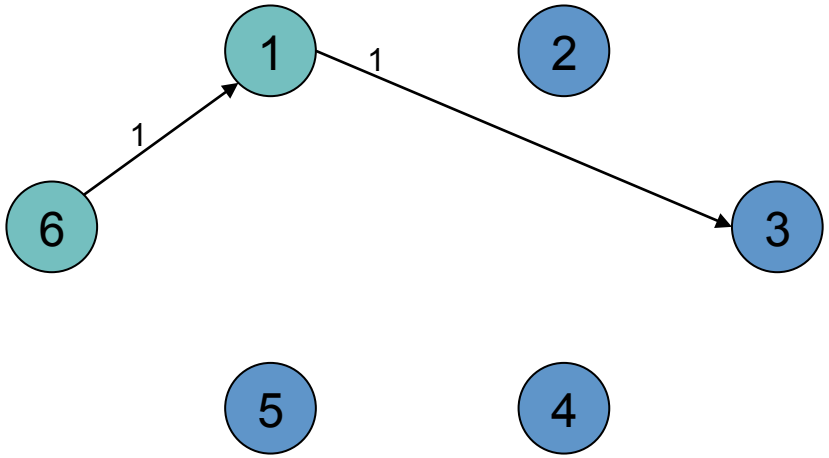


- **Distância Total =** 1

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

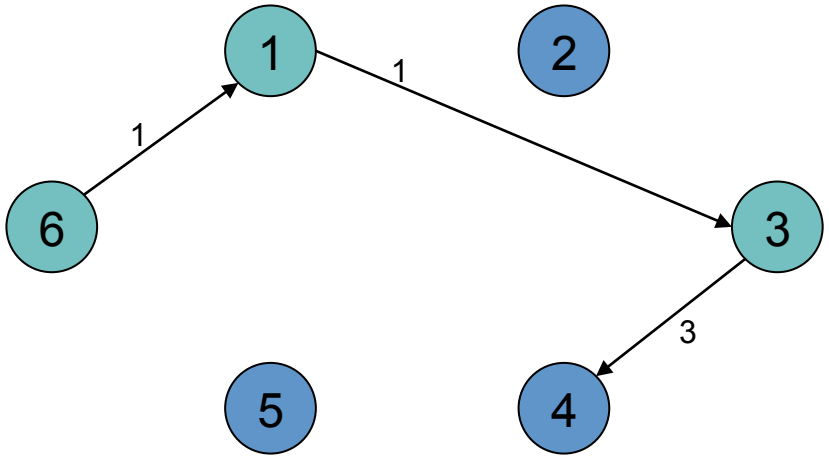


- **Distância Total =** 1 + 1 = 2

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

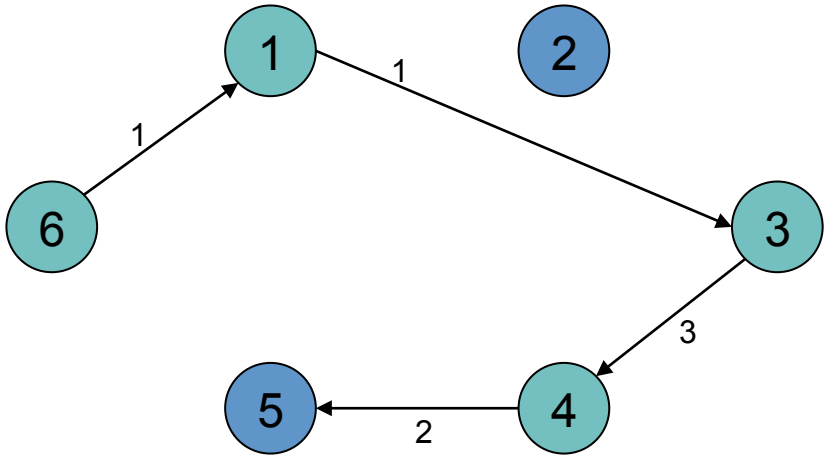


- **Distância Total =** 2 + 3 = 5

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

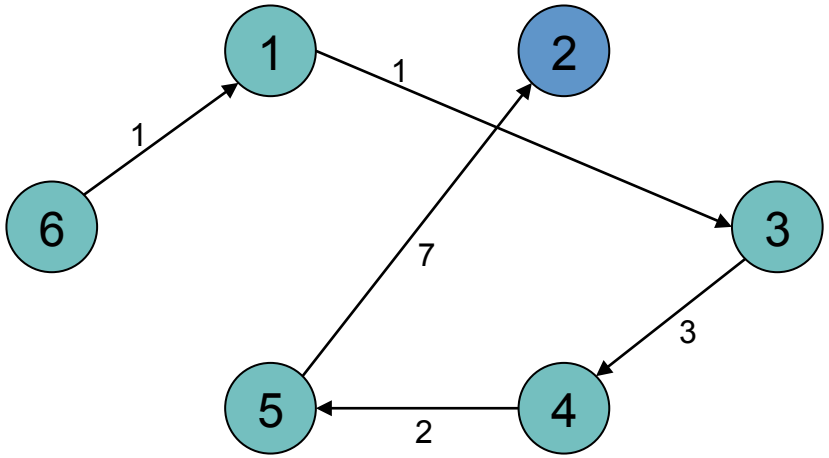


- **Distância Total =** 5 + 2 = 7

PCV – Vizinho mais Próximo
Exemplo - Passo 5

i	j	d_{ij}
5	2	7

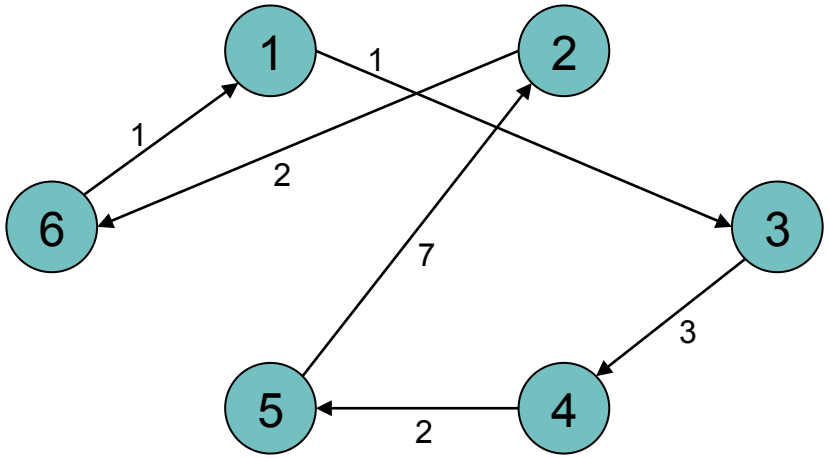
Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0



- **Distância Total =** 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

Heurísticas Construtivas - Exemplos

Problema de roteamento de veículos:

- Vizinho Mais Próximo;
- Economias/Inserção;

Heurísticas Construtivas: Vizinho Mais Próximo

Construir uma rota adicionando, a cada passo, a cidade mais próxima da última cidade inserida e que ainda não foi visitada.

Heurística do Vizinho Mais Próximo.

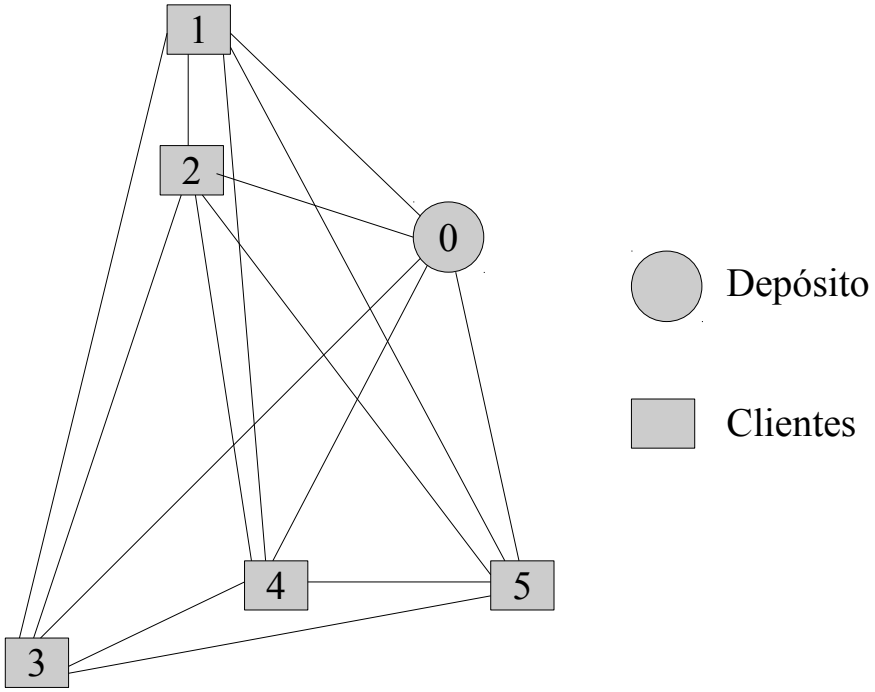
Passo 1: Escolha o vértice do depósito para começar.

Passo 2: Escolher um vértice ainda não visitado que seja o mais próximo do último vértice visitado que respeite a capacidade do veículo para inseri-lo na rota.

Passo 3: Se nenhum vértice pode ser adicionado na rota volte para o depósito.

Passo 4: Se todos os vértices já foram inseridos, PARE, caso contrário, volte ao Passo 2.

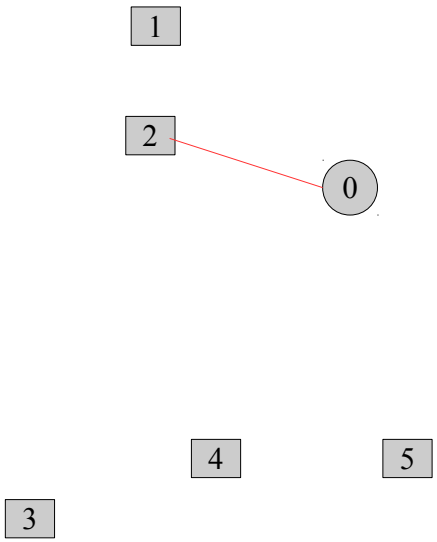
Exemplo da Aplicação do Vizinho Mais Próximo



$$c_{ij} = \begin{bmatrix} - & 12 & 9 & 17 & 13 & 10 \\ & - & 5 & 17 & 17 & 19 \\ & & - & 13 & 11 & 15 \\ & & & - & 7 & 10 \\ & & & & - & 6 \\ & & & & & - \end{bmatrix}, \quad d_j = [\ 31 \quad 22 \quad 18 \quad 29 \quad 37\ ], \quad Q = 100$$

Exemplo da Aplicação do Vizinho Mais Próximo

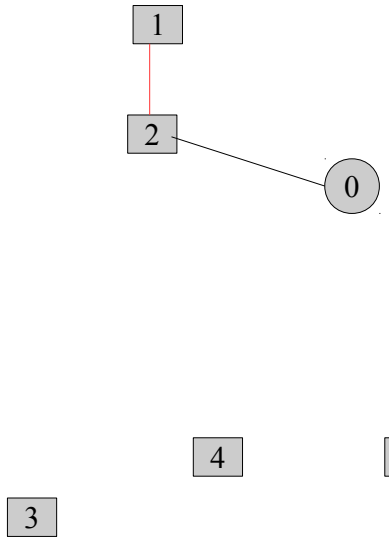
1ª Iteração



$$min\{c_{01}, c_{02}, c_{03}, c_{04}\, c_{0,5}\}$$
$$min\{12, 9, 17, 13, 10\}$$
$$c_{02}=9 \quad d_2=22$$

Rota	Custo	Demanda
0-2	9	22

2ª Iteração

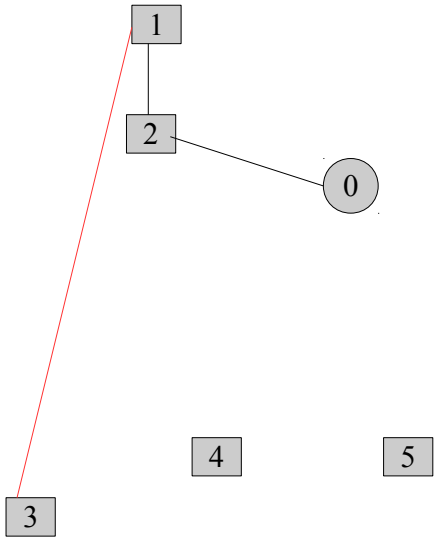


$$min\{c_{21}, c_{23}, c_{24}, c_{25}\}$$
$$min\{5, 13, 11, 15\}$$
$$c_{21}=5 \quad d_1=31$$

Rota	Custo	Demanda
0-2-1	14	53

Exemplo da Aplicação do Vizinho Mais Próximo

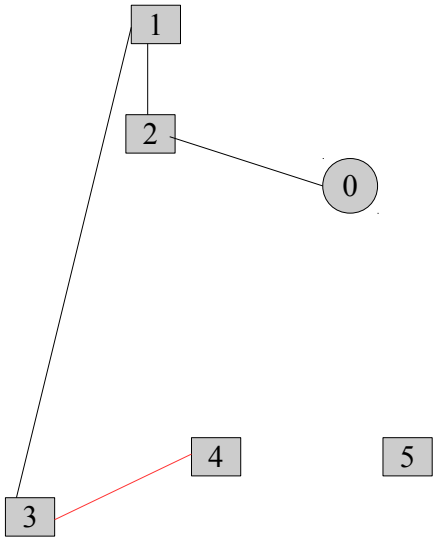
3ª Iteração



$$min\{c_{13}, c_{14}, c_{15}\}$$
$$min\{17, 17, 19\}$$
$$c_{13}=17 \quad d_3=18$$

Rota	Custo	Demanda
0-2-1-3	31	71

4ª Iteração

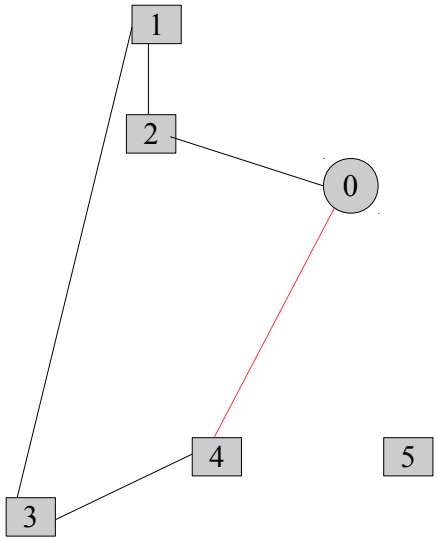


$$min\{c_{34}, c_{35}\}$$
$$min\{7, 10\}$$
$$c_{34}=7 \quad d_4=29$$

Rota	Custo	Demanda
0-2-1-3-4	38	100

Exemplo da Aplicação do Vizinho Mais Próximo

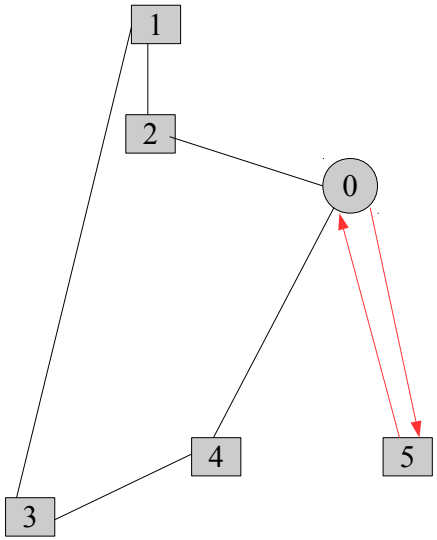
5ª Iteração



$c_{40}=13$

Rota	Custo	Demanda
0-2-1-3-4-0	51	100

6ª Iteração



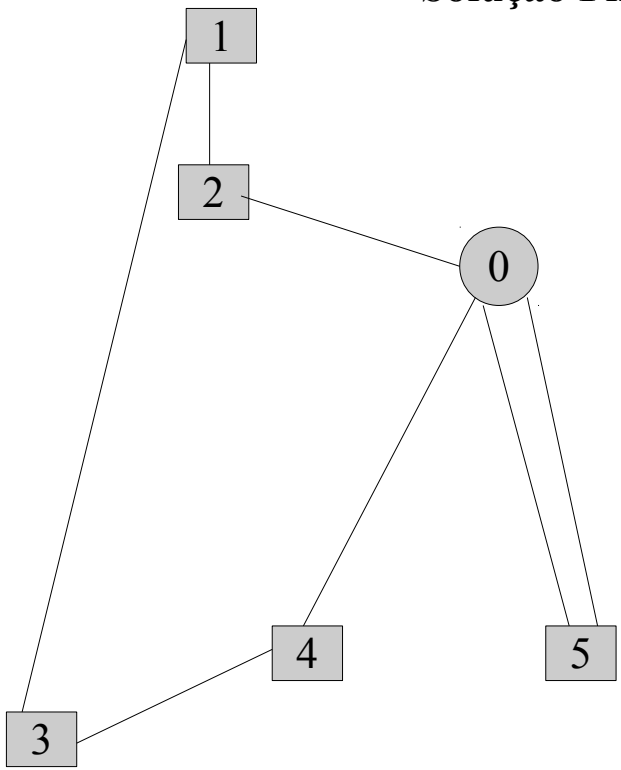
$c_{05}=13 \; + \; c_{50}=13$

$d_5=37$

Rota	Custo	Demanda
0-5-0	20	37

Exemplo da Aplicação do Vizinho Mais Próximo

Solução Final



Rota	Custo	Demanda
0-2-1-3-4-0	51	100
0-5-0	20	37
Total	71	137

Heurísticas Construtivas: Economias/Inserção

- Em cada passo, a solução atual é comparada com uma solução alternativa.
- Uma nova solução mais econômica (segundo algum critério) é produzida ou, uma solução que inclui uma demanda anteriormente não atendida é gerada.
- Exemplo: *Clarke and Wright*.

Algoritmo de *Clarke and Wright*

Algoritmo de *Clarke and Wright.*

Passo 1: No início, há uma rota para cada cliente. Cria-se n rotas $(0, i, 0)$ para $i = 1, \ldots, n$,.

%Em cada passo, duas rotas são concatenada de acordo com a economia proporciada.

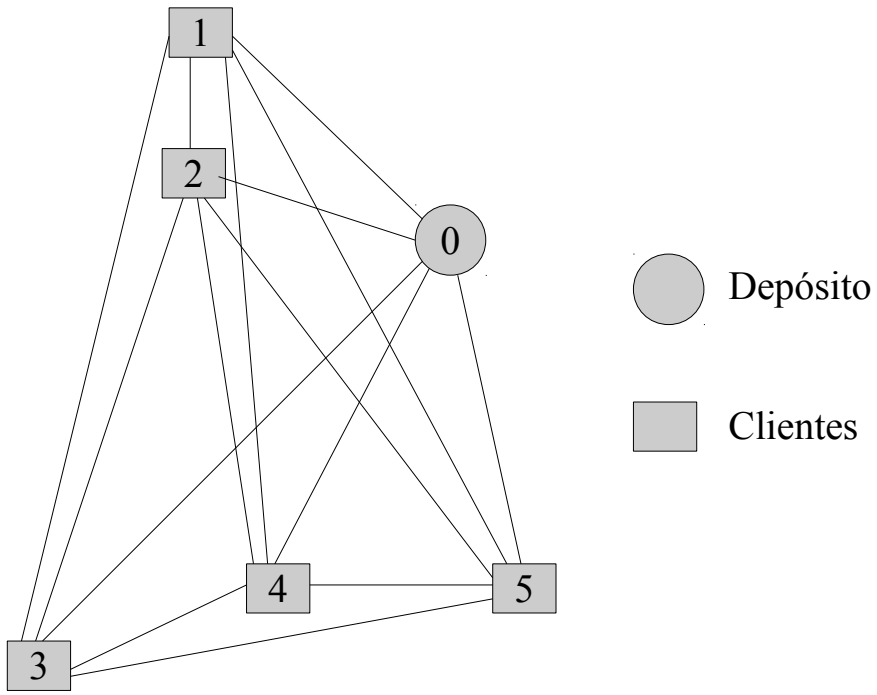
Passo 2: Calcular as economias $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ para $i, j = 1, \ldots, n$, e $i \neq j$.

Passo 3: Ordenar as economias em ordem não-crescente.

Passo 4: Considerar duas rotas contendo os clientes i e j em que $s_{ij} > 0$ e a capacidade do veículo é respeitada, adicionar o arco (i, j) à rota e excluir os arcos $(i, 0)$ e $(0, j)$. Repetir esse passo enquanto houver melhorias.

Passo 5: Se o número de rotas for limitado, repetir o Passo 4 e aceitar economias negativas.

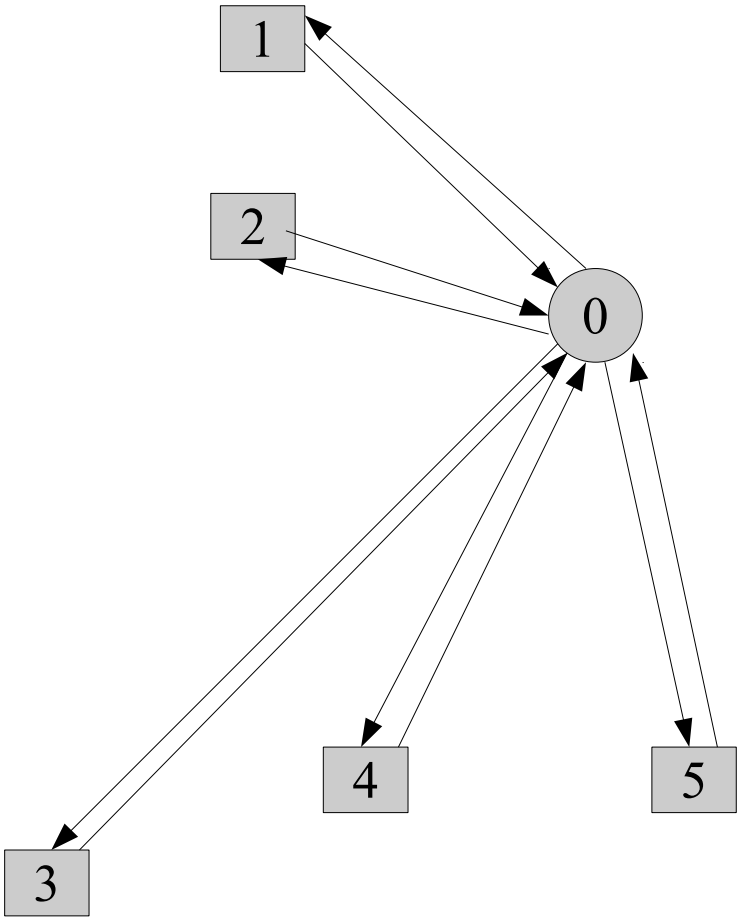
Exemplo da Aplicação do Algoritmo de *Clarke and Wright.*



$$c_{ij} = \begin{bmatrix} - & 12 & 9 & 17 & 13 & 10 \\ & - & 5 & 17 & 17 & 19 \\ & & - & 13 & 11 & 15 \\ & & & - & 7 & 10 \\ & & & & - & 6 \\ & & & & & - \end{bmatrix}, \quad d_j = [\ 31 \quad 22 \quad 18 \quad 29 \quad 37\ ], \quad Q = 100$$

Exemplo da Aplicação do Algoritmo de *Clarke and Wright.*

Passo 1: Uma rota para cada cliente;



Passo 2: Matriz de economia;

$$s_{ij} = c_{i0} + c_{0j} - c_{ij}$$

s_{ij}	1	2	3	4	5
1	-	16	12	8	3
2		-	9	5	0
3			-	13	13
4				-	17

Passo 3: Economias ordenadas;

1: [4, 5] **6:** [2, 3]

2: [1, 2] **7:** [1, 4]

3: [3, 4] **8:** [2, 4]

4: [3, 5] **9:** [1, 5]

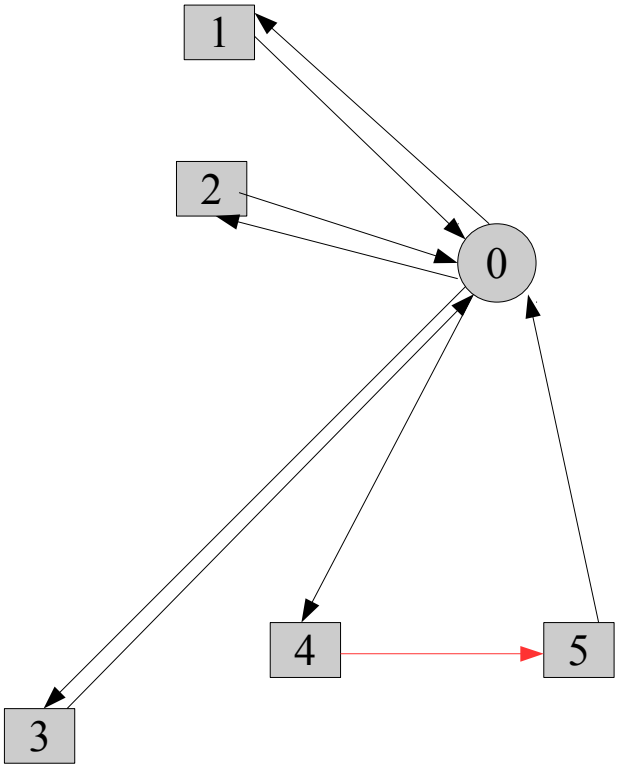
5 [1, 3] **10:** [2, 5]

Exemplo da Aplicação do Algoritmo de *Clarke and Wright.*

Arco [4, 5] : $i = 4$ e $j = 5$

Juntar os ciclos 0-4-0 e 0-5-0: $(0, j) = (0,5)$ e $(i, 0) = (4, 0)$.

Demanda: $29 + 37 = 66 < 100$.

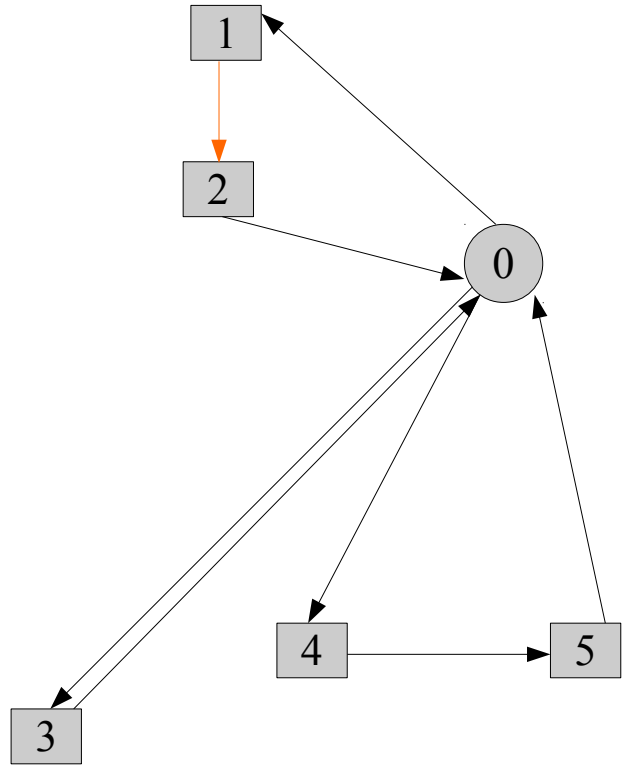


Custo: 29 Demanda: 66
Rota: 0-4-5-0

Arco [1, 2] : $i = 1$ e $j = 2$

Juntar os ciclos 0-1-0 e 0-2-0: $(0, j) = (0,2)$ e $(i, 0) = (1, 0)$.

Demanda: $31 + 22 = 53 < 100$.



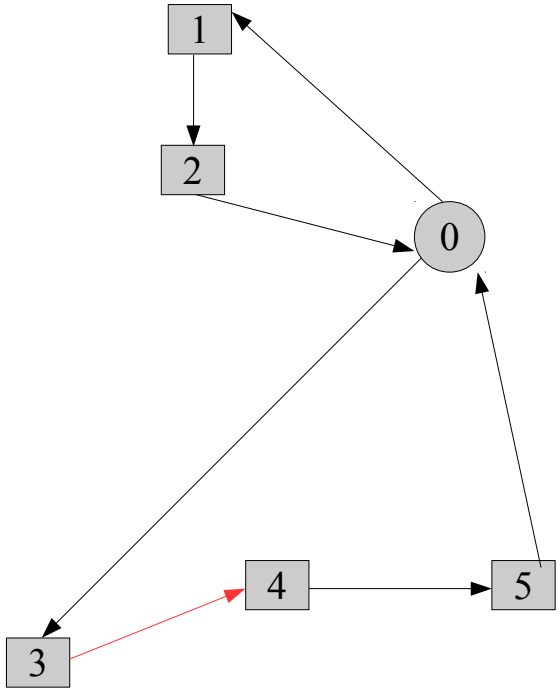
Custo: 26 Demanda: 53
Rota: 0-1-2-0

Exemplo da Aplicação do Algoritmo de *Clarke and Wright.*

Arco [3, 4] : $i = 3$ e $j = 4$

Juntar os ciclos 0-3-0 e 0-4-5-0: $(0, j) = (0,4)$ e $(i, 0) = (3, 0)$.

Demanda: 66 + 18 = 84 < 100.



Custo: 48 Demanda: 84
Rota: 0-3-4-5-0

Arco [3, 5] : $i = 3$ e $j = 5$

Ambos os vértices estão na mesma rota.

Arco [1, 3] : $i = 1$ e $j = 3$

Juntar os ciclos 0-1-2-0 e 0-3-4-5-0: $(0, j) = (0,3)$ e $(i, 0) \neq (1, 0)$.

Arco [2, 3] : $i = 2$ e $j = 3$

Juntar os ciclos 0-1-2-0 e 0-3-4-5-0: $(0, j) = (0,3)$ e $(i, 0) = (2, 0)$.

Demanda: 84 + 53 = 137 > 100.

Arco [1, 4] : $i = 1$ e $j = 4$

Juntar os ciclos 0-1-2-0 e 0-3-4-5-0: $(0, j) = (0,4)$ e $(i, 0) \neq (1, 0)$.

Arco [2, 4] : $i = 1$ e $j = 4$

Juntar os ciclos 0-1-2-0 e 0-3-4-5-0: $(0, j) \neq (0,4)$ e $(i, 0) = (2, 0)$.

Arco [1, 5] : $i = 1$ e $j = 5$

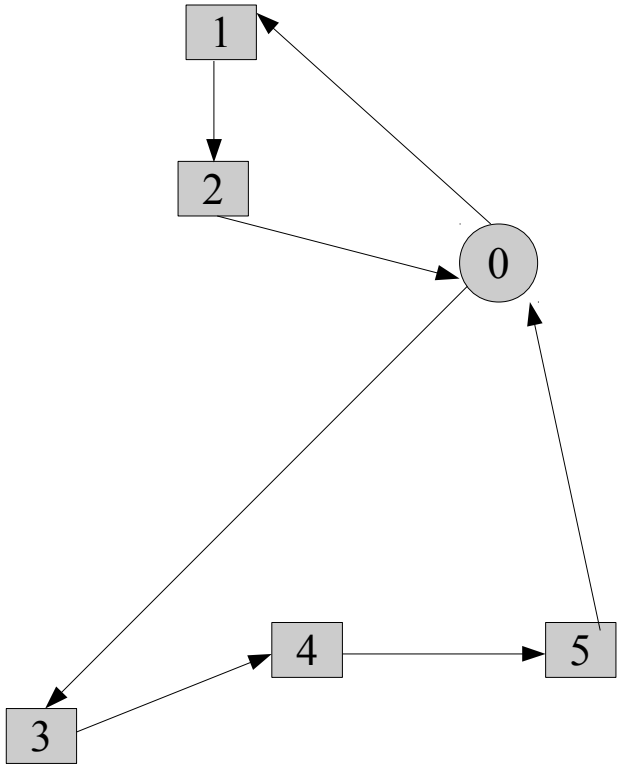
Juntar os ciclos 0-1-2-0 e 0-3-4-5-0: $(0, j) \neq (0,5)$ e $(i, 0) \neq (1, 0)$.

Arco [2, 5] : $i = 2$ e $j = 5$

Juntar os ciclos 0-1-2-0 e 0-3-4-5-0: $(0, j) \neq (0,5)$ e $(i, 0) = (2, 0)$.

Exemplo da Aplicação do Algoritmo de *Clarke and Wright.*

Solução Final



Rota: 0-3-4-5-0
Custo: 48
Demanda: 84

Rota: 0-1-2-0
Custo: 26
Demanda: 53

Custo Total: 74

Heurísticas de Busca Local

- Um algoritmo de busca local define, para cada solução, uma vizinhança composta por um conjunto de soluções com características "muito próximas".
- Começa-se com uma solução inicial S obtida, por exemplo, por uma heurística construtiva;
- Uma nova solução factível $S^{'}$ é gerada através de uma operação chamada movimento;
- A nova solução $S^{'}$ é chamada vizinha de S;
- A busca local visa explorar a vizinhança de uma dada solução por meio de operações que realizaram movimentos sobre a soluções corrente;
- Seleciona-se uma solução vizinha $S^{'}$ que seja melhor que a solução corrente S;
- A busca prossegue até que a vizinhança não contenha solução melhor que a corrente (ótima local.)

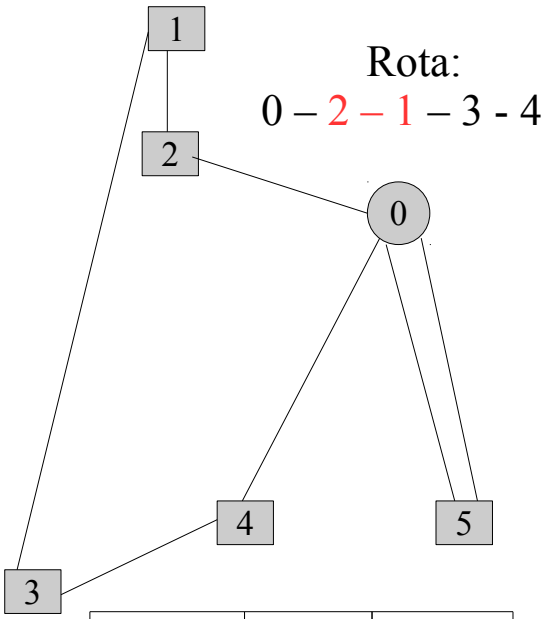
Heurísticas de Busca Local - Ex. PRV

- Ponto chave definir o tipo de movimento:
  - Troca: busca realizar movimentos troca entre dois ou mais vértices da solução;
  - Inserção: consiste em inserir um ou mais elementos em outra posição;
  - r-opt: realiza movimentos de troca entre arestas.

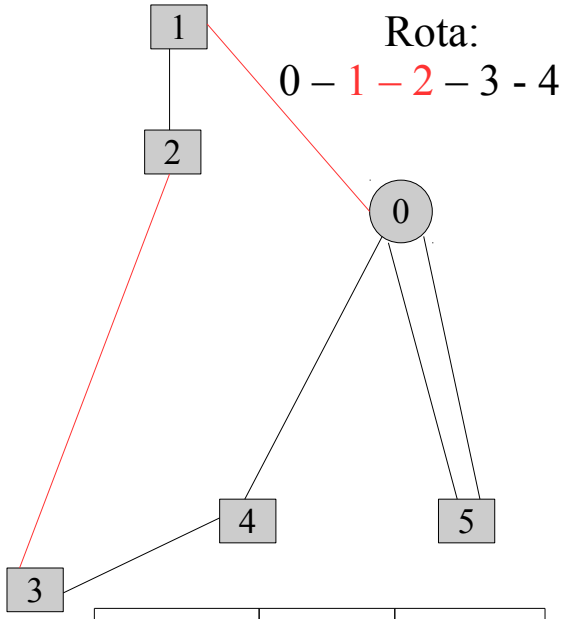
Movimento de Troca - Ex. PRV

- Seleciona-se dois vértices para serem trocados na ordem em que vão ser visitados;
- Os clientes i e j terão suas posições trocadas.

Trocar os clientes 1 e 2



Rota	Custo	Demanda
0-2-1-3-4-0	51	100
0-5-0	20	37
Total	71	137



Rota	Custo	Demanda
0-1-2-3-4-0	48	100
0-5-0	20	37
Total	68	137

Referências

- MARTÍ, R.; REINELT, G. Heuristic methods.The linear ordering problem: exact and heuristic methods in combinatorial optimization. Springer-Verlag, Berlin, Heidelberg, 2011. cap. 2, p. 17?40.