Pesquisa Operacional Aplicada à Mineração

Módulo de Otimização – Parte II-b

Prof. Marcone J. F. Souza
Prof. Túlio A. M. Toffolo

marcone.freitas@yahoo.com.br | tulio@toffolo.com.br

Departamento de Computação
Universidade Federal de Ouro Preto

Pesquisa Operacional Aplicada à Mineração

- **Prof. Marcone Jamilson Freitas Souza**
  Departamento de Computação
  Universidade Federal de Ouro Preto
  www.decom.ufop.br/prof/marcone
  marcone.freitas@yahoo.com.br

- **Prof. Túlio Ângelo Machado Toffolo**
  Departamento de Computação
  Universidade Federal de Ouro Preto
  www.decom.ufop.br/toffolo
  tulio@toffolo.com.br

Roteiro

- Problema de sequenciamento em máquinas paralelas
- Problema da Seleção de Projetos
- Problema do Caixeiro Viajante
- Heurísticas computacionais para otimização
  - Conceitos básicos
  - Heurísticas construtivas
  - Heurísticas clássicas de refinamento
  - Metaheurísticas

PROBLEMA DE SEQUENCIAMENTO EM MÁQUINAS PARALELAS E IDÊNTICAS

Sequenciamento em máquinas paralelas e idênticas

- Há um conjunto de máquinas (Maquinas) e um conjunto (Tarefas) de tarefas a serem executadas nessas máquinas.
- Cada tarefa pode ser executada em qualquer ordem e em qualquer máquina, e estão disponíveis desde o instante zero
- Existe um tempo t_j para processar a tarefa j
- Uma vez iniciada a execução de uma tarefa, ela não pode ser interrompida
- Deve-se minimizar o tempo de processamento de todas as tarefas (makespan)

Sequenciamento em máquinas paralelas e idênticas

Dados de entrada:

- t_j = tempo de processamento da tarefa j

Variáveis de decisão:

- x_{ij} = 1 se a tarefa j é alocada à máquina i e 0, c.c.
- C_{max} = instante de término da máquina mais sobrecarregada

Sequenciamento em máquinas paralelas e idênticas

- Minimizar o tempo de término da máquina mais sobrecarregada

$$\min C_{\max}$$

- Cada tarefa j deve ser alocada a uma única máquina i

$$\sum_{i \in Maquinas} x_{ij} = 1 \quad \forall j \in Tarefas$$

Sequenciamento em máquinas paralelas e idênticas

- C_{max} deve ser maior ou igual a cada tempo de término de máquina, representando assim o maior tempo de término

$$C_{\max} \geq \sum_{j \in Tarefas} t_j x_{ij} \; \forall i \in Maquinas$$

Sequenciamento em máquinas paralelas e idênticas

- Modelo completo:

$$\min C_{\max}$$

$$\sum_{i \in Maquinas} x_{ij} = 1 \quad \forall j \in Tarefas$$

$$C_{\max} \geq \sum_{j \in Tarefas} t_j x_{ij} \forall i \in Maquinas$$

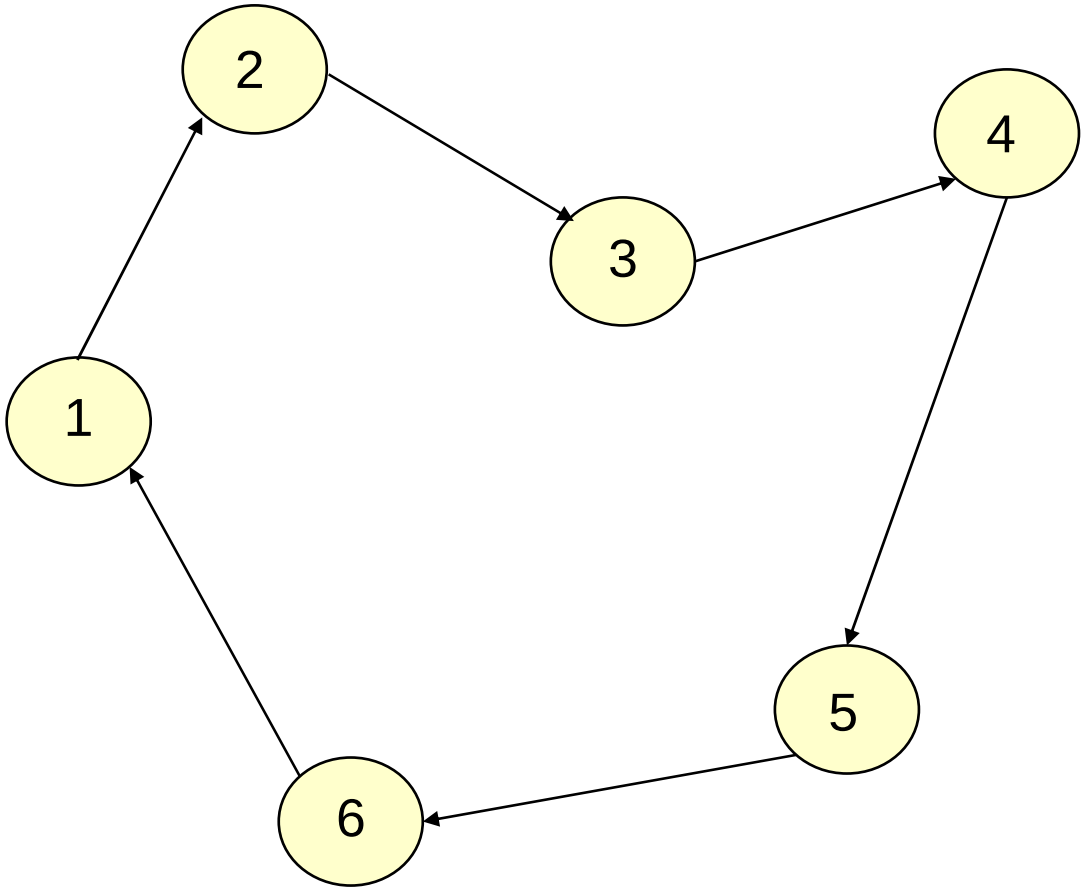
$$x_{ij} \in \{0,1\} \; \forall i \in Maquinas, \forall j \in Tarefas$$

PROBLEMA DO CAIXEIRO VIAJANTE

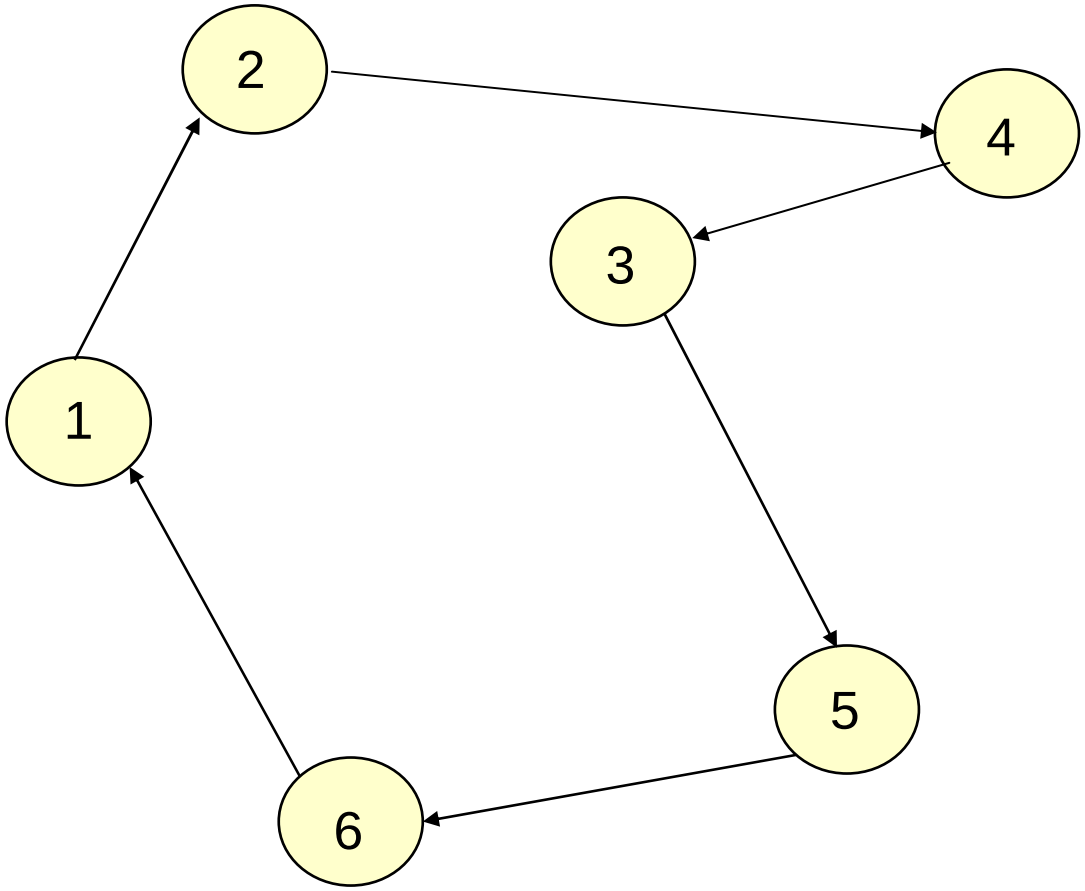
Problema do Caixeiro Viajante

- 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



Problema do Caixeiro Viajante



Problema do Caixeiro Viajante

Dados de entrada:

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

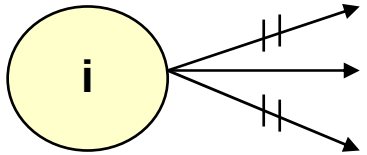
Variáveis de decisão:

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

Problema do Caixeiro Viajante

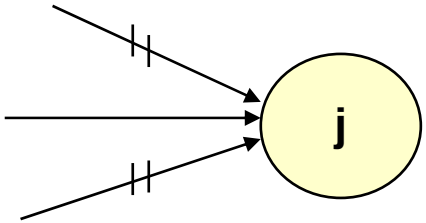
- De cada cidade i só sai uma aresta:

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



- A cada cidade j só chega uma aresta:

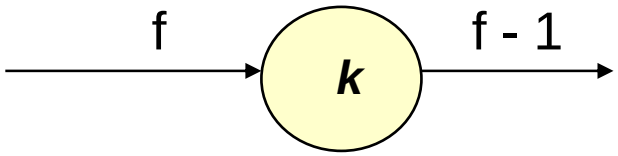
$$\sum_{i \in Cidades} x_{ij} = 1 \quad \forall j \in Cidades$$



Problema do Caixeiro Viajante

- O fluxo que chega a uma cidade i menos o que sai é igual a uma unidade:

$$\sum_{i \in Cidades} f_{ik} - \sum_{j \in Cidades} f_{kj} = 1 \quad \forall k \in Cidades \mid k \neq 1$$



Problema do Caixeiro Viajante

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

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

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

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

Problema do Caixeiro Viajante

- 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

- Para dimensões mais elevadas, a resolução por modelos de programação matemática é proibitiva em termos de tempos computacionais
- PCV é da classe NP-difícil: ainda não existem algoritmos exatos que o resolvam em tempo polinomial
- À medida que *n* cresce, o tempo de execução cresce exponencialmente

Problema do Caixeiro Viajante

- 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

PROBLEMA DO CAIXEIRO VIAJANTE

Heurísticas

- Construtivas
  - Consistem em construir uma solução passo a passo, elemento por elemento
- De refinamento
  - Consistem em efetuar modificações na solução construída, de forma a tentar melhorá-la

Heurística de construção gulosa

- 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

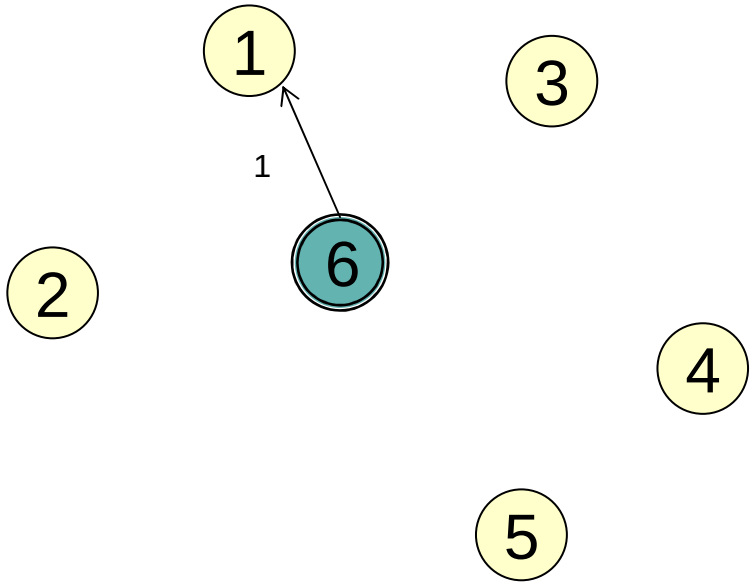
Heurísticas construtivas

- Vizinho mais próximo
  - Idéia central: Construir uma rota passo a passo, adicionando à solução corrente a cidade mais próxima (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	2
4	4	9	3	0	2	3
5	9	7	8	2	0	2
6	1	2	2	3	2	0

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

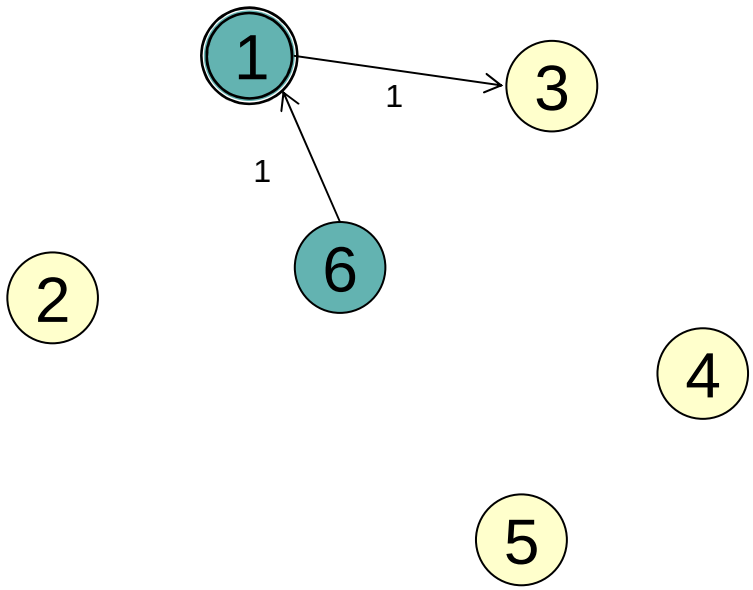


Dist = 1

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

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

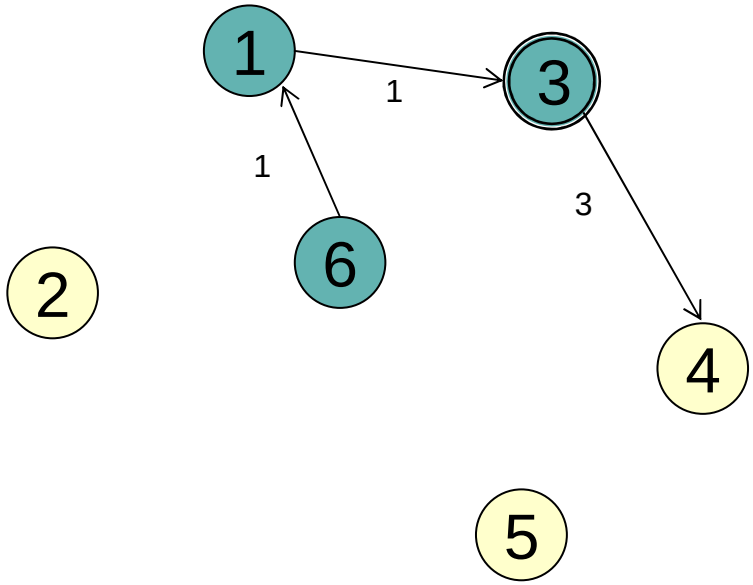


Dist = 1 + 1 = 2

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

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

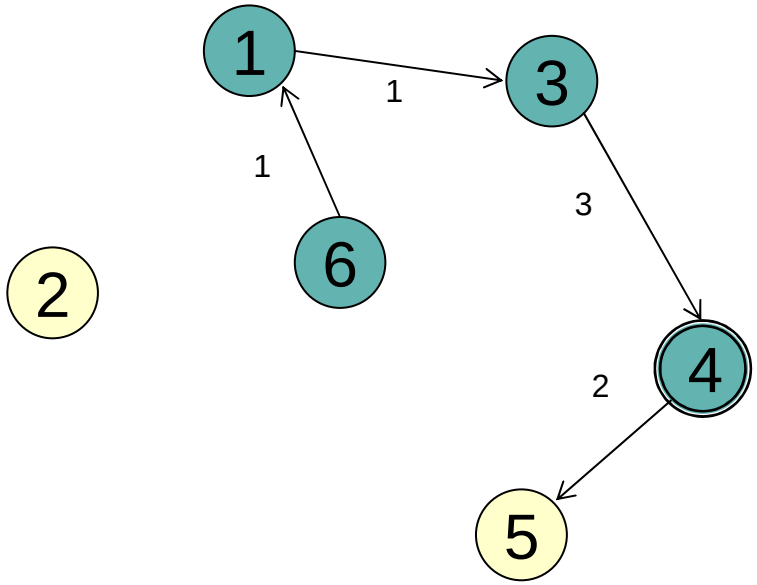


Dist = 2 + 3 = 5

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

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

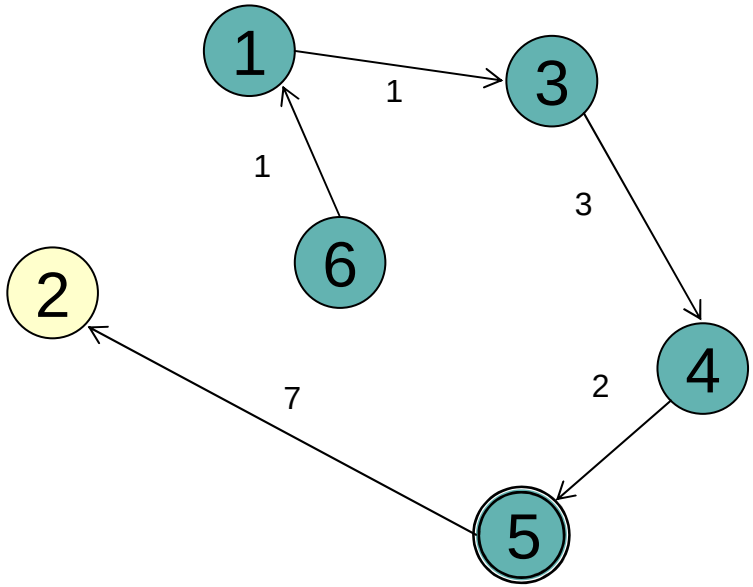


Dist = 5 + 2 = 7

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

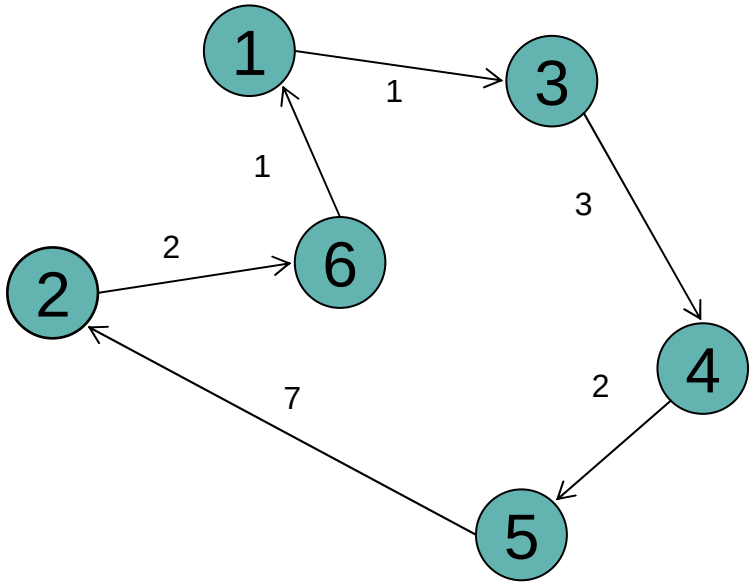
i	j	d_{ij}
5	**2**	7



Dist = 7 + 7 = 14

PCV – Vizinho mais Próximo :: 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	2
4	4	9	3	0	2	3
5	9	7	8	2	0	2
6	1	2	2	3	2	0



Dist = 14 + 2 = 16

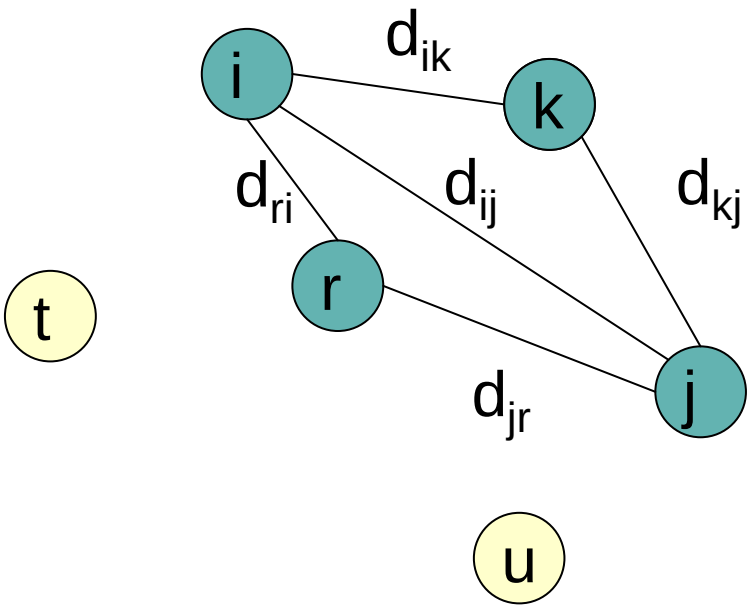
S = (6,1,3,4,5,2)

Heurística da Inserção Mais Barata

- 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

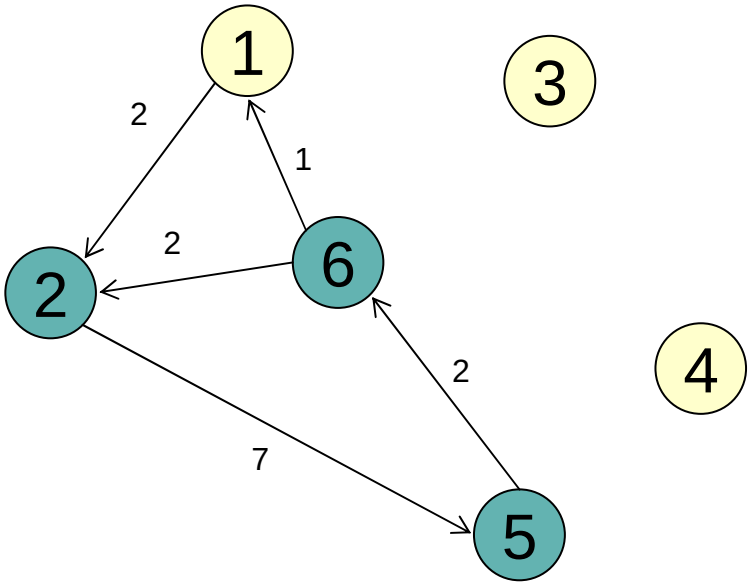
Heurística da Inserção Mais Barata

Custo da inserção = $d_{ik} + d_{kj} - d_{ij}$



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



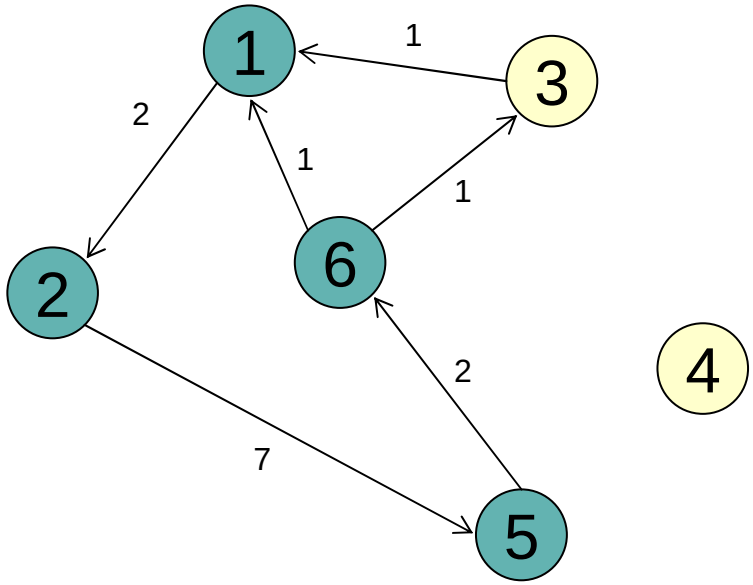
i	k	j	$d_{ik} + d_{kj} - d_{ij}$
6	**1**	**2**	1 + 2 – 2 = 1
6	**3**	**2**	2 + 5 – 2 = 5
6	**4**	**2**	3 + 9 – 2 = 10
2	**1**	**5**	2 + 9 – 7 = 4
2	**3**	**5**	5 + 8 – 7 = 6
2	**4**	**5**	9 + 2 – 7 = 4
5	**1**	**6**	9 + 1 – 2 = 8
5	**3**	**6**	8 + 6 – 2 = 12
5	**4**	**6**	2 + 3 – 2 = 3

Dist = 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	2
4	4	9	3	0	2	3
5	9	7	8	2	0	2
6	1	2	2	3	2	0

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

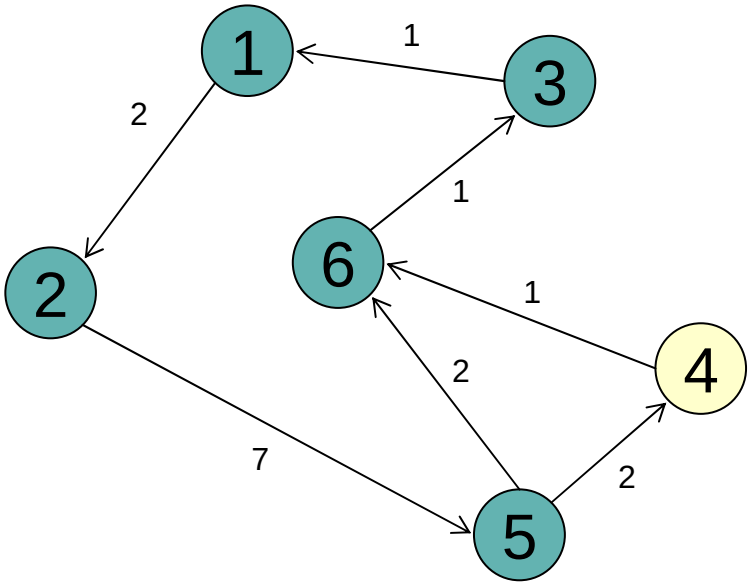


Dist = 12 + 2 = 14

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

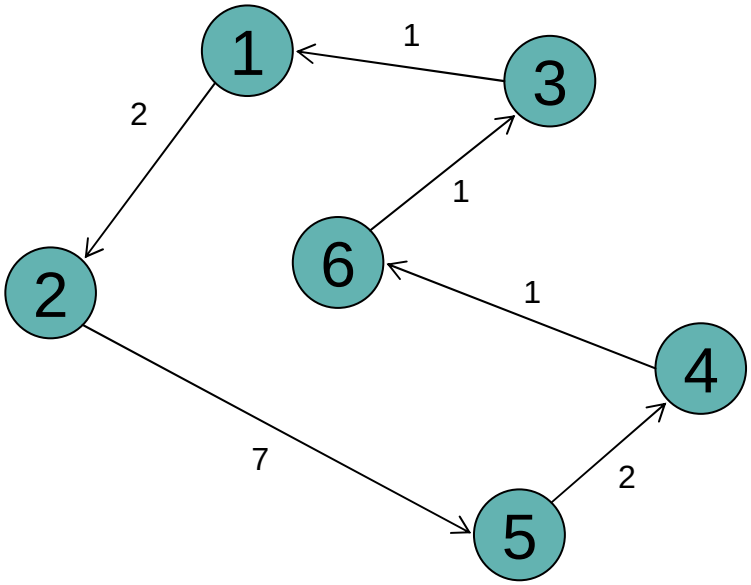
i	k	j	$d_{ik} + d_{kj} - d_{ij}$
6	**4**	**3**	3 + 3 – 2 = 4
3	**4**	**1**	3 + 4 – 1 = 6
1	**4**	**2**	4 + 9 – 2 = 11
2	**4**	**5**	9 + 2 – 7 = 4
5	**4**	**6**	2 + 3 – 2 = 3



Dist = 14 + 3 = 17

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

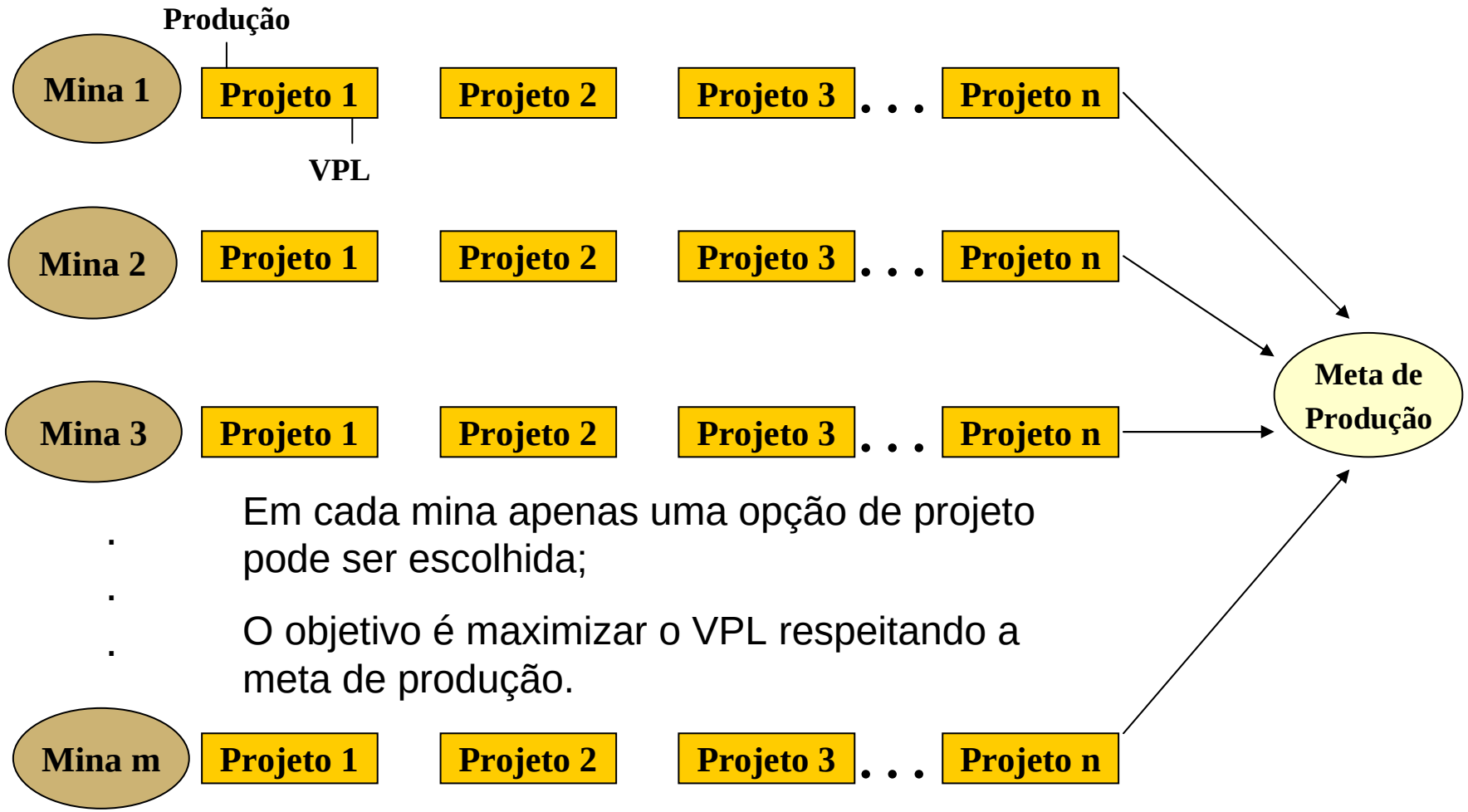


Dist = 17

S = (6 3 1 2 5 4)

PROBLEMA DA SELEÇÃO DE PROJETOS

Seleção de Projetos Concorrentes



Seleção de Projetos Concorrentes

Opção	Produção	VPL ($\$x10^6$)	Custo/t.	Observações:
1	40.00	884.42	4.59	Caso Base.
2	43.00	929.56	4.36	Aumento Capacidade.
3	43.00	922.04	5.08	Aumento Capacidade c/risco operacional.
4	43.00	885.11	5.16	Aumento Capacidade c/ aumento custo.
5	40.00	857.75	4.39	Caso Base com variação de risco.
6	40.00	874.16	4.61	Caso Base com variação de risco.
7	40.00	1000.00	4.42	Caso Base com variação de risco.
8	40.00	857.07	4.56	Caso Base com variação de risco.
9	40.00	804.41	4.63	Caso Base com variação de risco.
10	42.00	967.17	4.70	Aumento Capacidade 2.
11	42.00	909.04	4.75	Aumento Capacidade 2 c/ risco.
12	42.00	843.39	4.84	Aumento Capacidade 2 c/ risco.
13	48.00	972.64	3.88	Aproveitamento Minério Marginal.
14	48.00	935.71	3.90	Ídem c/ variação de investimento.
15	45.00	930.93	4.14	Ídem c/ variação de produção.
16	45.00	897.42	4.16	Ídem c/ variação de investimento.

Seleção de Projetos Concorrentes

Exemplo de VPL ($x10^6):

	1	*2*	*3*	...	*16*
1	884.42	929.56	922.04	...	897.42
2	177.28	149.92	199.85	...	247.04
...	...	...	...	...	...
7	111.63	106.84	103.42	...	126.67

Exemplo de Produção (Mt):

	1	*2*	*3*	...	*16*
1	40	43	43	...	45
2	10	10	10	...	12.5
...	...	...	...	...	...
7	2	2	2	...	4

Seleção de Projetos Concorrentes

Dados de entrada:

- Minas: conjunto de minas;
- Projetos: conjunto de opções de projeto em cada mina;
- VPL_{ij}: Valor Presente Líquido do projeto j para a mina i;
- $prod_{ij}$: produção do projeto j na mina i;
- M: meta de produção;
- Pfalta: penalidade por produção inferior à meta estabelecida;
- Pexc: penalidade por produção superior à meta estabelecida.

Seleção de Projetos Concorrentes

Variável de Decisão:

- $x_{ij} = \begin{cases} 1 \text{ caso a mina i opere com o projeto j} \\ 0 \text{ caso contrário} \end{cases}$

Seleção de Projetos Concorrentes

$$\max \sum_{i \in Minas} \sum_{j \in Projetos} VPL_{ij} x_{ij} - Pfalta \times dnp - Pexc \times dpp$$

$$\sum_{j \in Projetos} x_{ij} = 1 \; \forall i \in Minas$$

$$\sum_{i \in Minas} \sum_{j \in Projetos} prod_{ij} x_{ij} - dpp + dnp = M$$

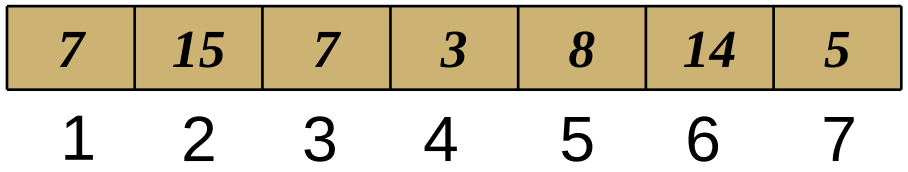
$$x_{ij} \in \{0,1\} \; \forall i \in Minas, \; \forall j \in Projetos$$

HEURÍSTICA

PROBLEMA DA SELEÇÃO DE PROJETOS

Seleção de Projetos Concorrentes

Representação de uma solução:



*Obs.: Com esta representação, a restrição de que em cada mina um projeto tem que ser implementado é **automaticamente** satisfeita

Heurísticas de refinamento

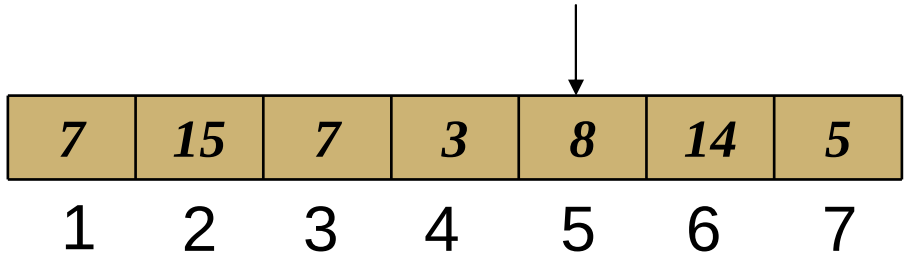
Princípio de funcionamento:

- Partir de uma solução inicial qualquer
- Caminhar, a cada iteração, de vizinho para vizinho de acordo com a definição de vizinhança adotada, tentando **melhorar** a solução construída

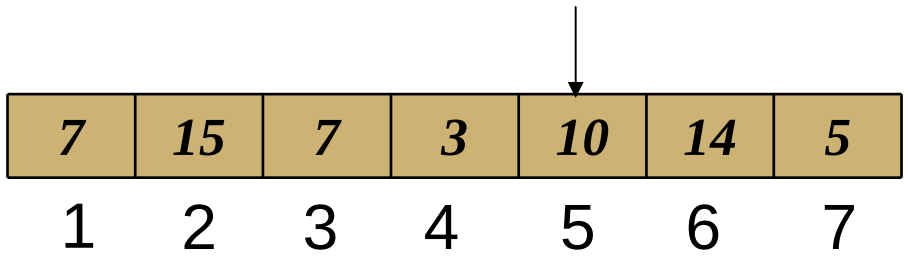
Seleção de Projetos Concorrentes

Exemplo de vizinhança:

Solução *s*:



Exemplo de vizinho *s'*: gerado a partir de s, substituindo-se o projeto implementado em uma mina por outro



Método da descida/subida (Descent/Uphill Method)

- Parte de uma solução inicial qualquer
- A cada passo analisa **todos** os possíveis vizinhos
- Move somente para o vizinho que representa uma **melhora** no valor atual da função de avaliação.
  - Em problemas de minimização, um vizinho s' é melhor que s quando $f(s') < f(s)$). Neste caso, s' passa a ser a nova solução corrente
- O método é interrompido quando um ótimo local (com respeito à definição de vizinhança) é encontrado

METAHEURÍSTICAS

Metaheurísticas

- Métodos heurísticos, de caráter geral, dotados de mecanismos para escapar das armadilhas dos ótimos locais
- Princípio básico: aceitar soluções de piora
- Podem ser baseados em Busca Local ou Busca Populacional.
- Os métodos baseados em Busca Local são fundamentados na noção de vizinhança:
  - Dada uma solução s, diz-se que s' é um vizinho de s, se s' é obtido de s a partir de um movimento m, isto é: $s' \leftarrow s \oplus m$
  - A estrutura de vizinhança varia de acordo com o problema tratado
- Os métodos baseados em Busca Populacional partem de um conjunto de soluções, aplicando sobre estes operadores que visam à melhoria desse conjunto.

Metaheurísticas

- Exemplos de metaheurísticas:
  - de busca local:
    - Busca Tabu
    - *Simulated Annealing*
    - *Iterated Local Search* (ILS)
    - *Variable Neighborhood Search* (VNS)
  - de busca populacional:
    - Algoritmos Genéticos
    - Colônia de Formigas
    - Otimização por Nuvem de Partículas

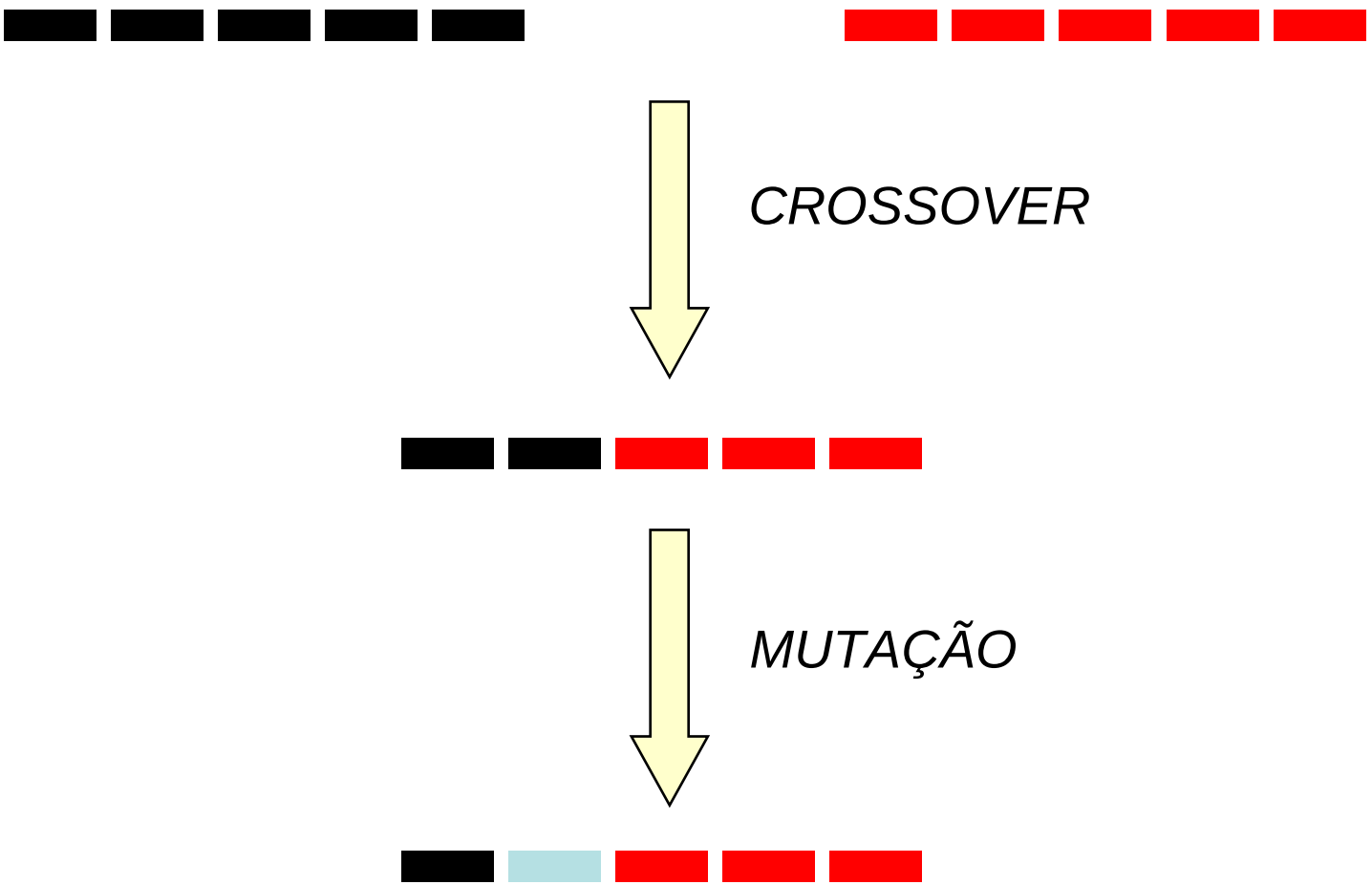
Algoritmos Genéticos

- Os AGs fazem uma analogia com processos naturais de evolução:
- Dada uma população, os indivíduos com características genéticas melhores têm maiores chances de sobrevivência e de produzirem filhos cada vez mais aptos, enquanto indivíduos menos aptos tendem a desaparecer
- As características dos indivíduos, registradas em seus genes, são transmitidas para seus descendentes e tendem a propagar-se por novas gerações

Algoritmos Genéticos

- Características dos descendentes são parcialmente herdadas de seus pais (**Crossover**) e parcialmente de novos genes criados durante o processo de reprodução (**Mutação**)

Operadores genéticos



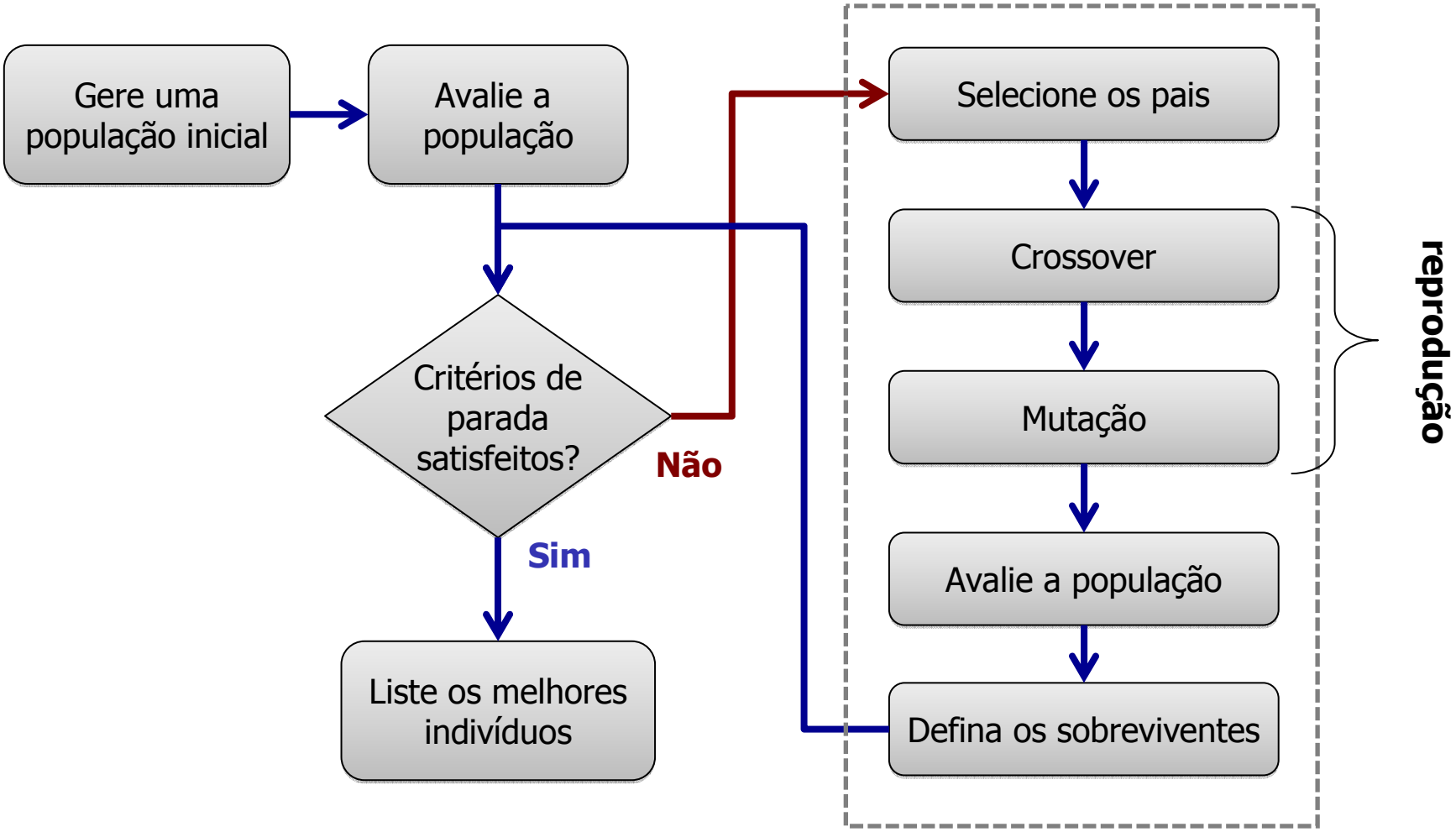
Algoritmos Genéticos

- O objetivo de um AG é tentar melhorar as qualidades genéticas de uma população através de um processo de renovação iterativa das populações

Relação entre AG e Problema de Otimização

AG	Problema de Otimização
Indivíduo	Solução de um problema
População	Conjunto de soluções
Cromossomo	Representação de uma solução
Gene	Parte da representação de uma solução
Alelos	Valores que uma variável pode assumir
Crossover / Mutação	Operadores de busca

Algoritmos Genéticos



Avaliação de cromossomos

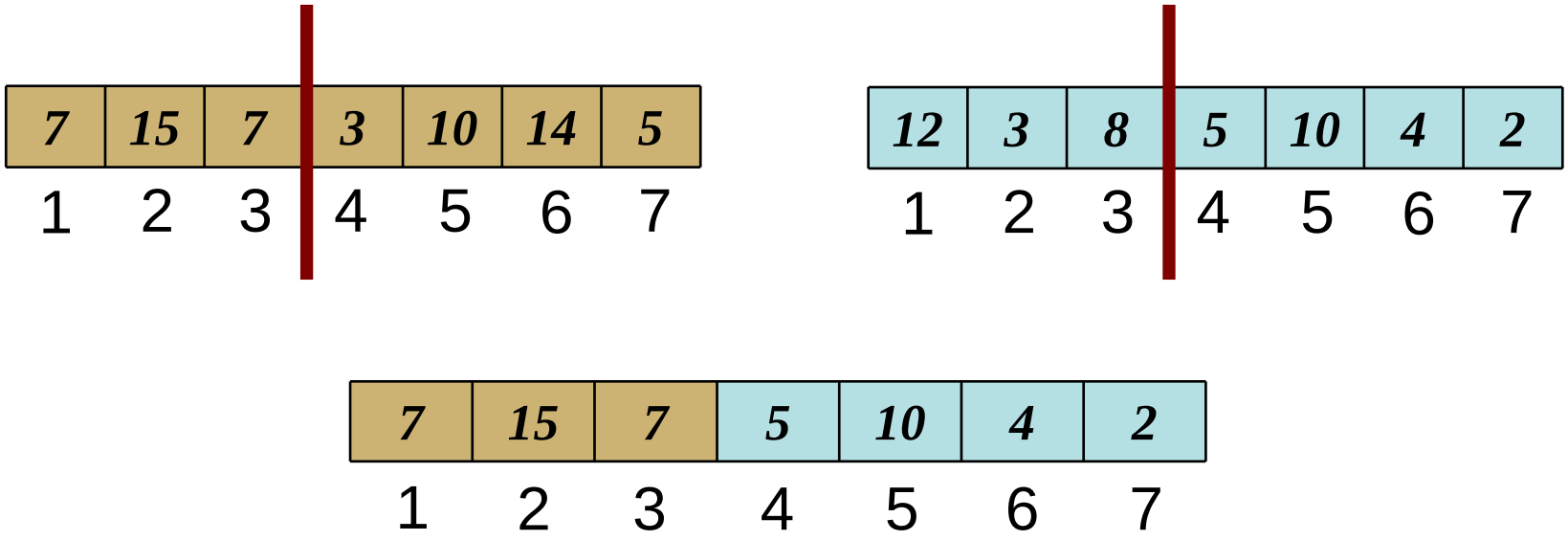
- Feita pela função de aptidão (**fitness**)
- Em um problema de maximização pode ser a própria função objetivo
- Em um problema de minimização pode ser o inverso da função objetivo

Fase de seleção

- Binary tournament selection:
  - Selecionar dois indivíduos aleatoriamente
  - O primeiro pai é o indivíduo com maior aptidão
  - Selecionar, aleatoriamente, outros dois pais
  - O segundo pai é o indivíduo com maior aptidão nessa nova seleção
- Aleatório
- Roleta russa

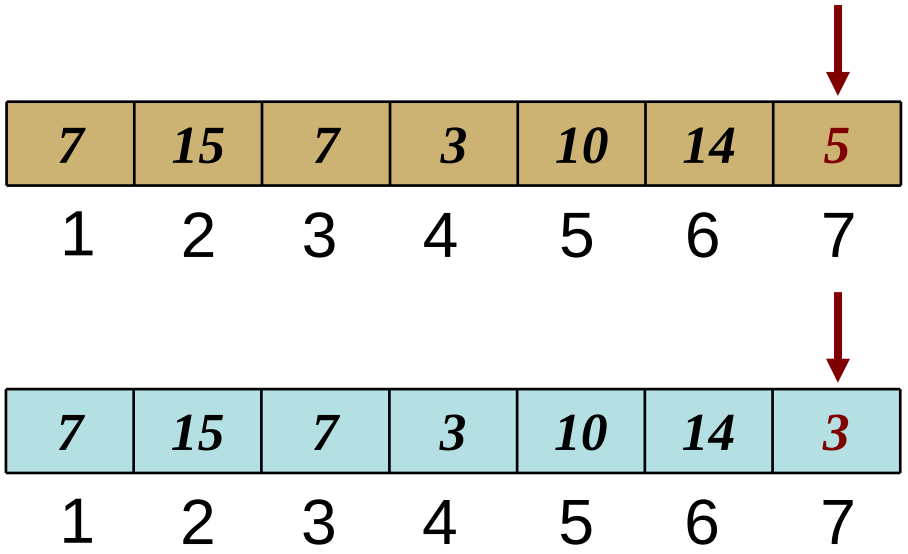
Fase de reprodução

- Operador **crossover** clássico (*one point crossover*):
- Descendentes são formados a partir da reunião de segmentos de cada pai:



Fase de reprodução

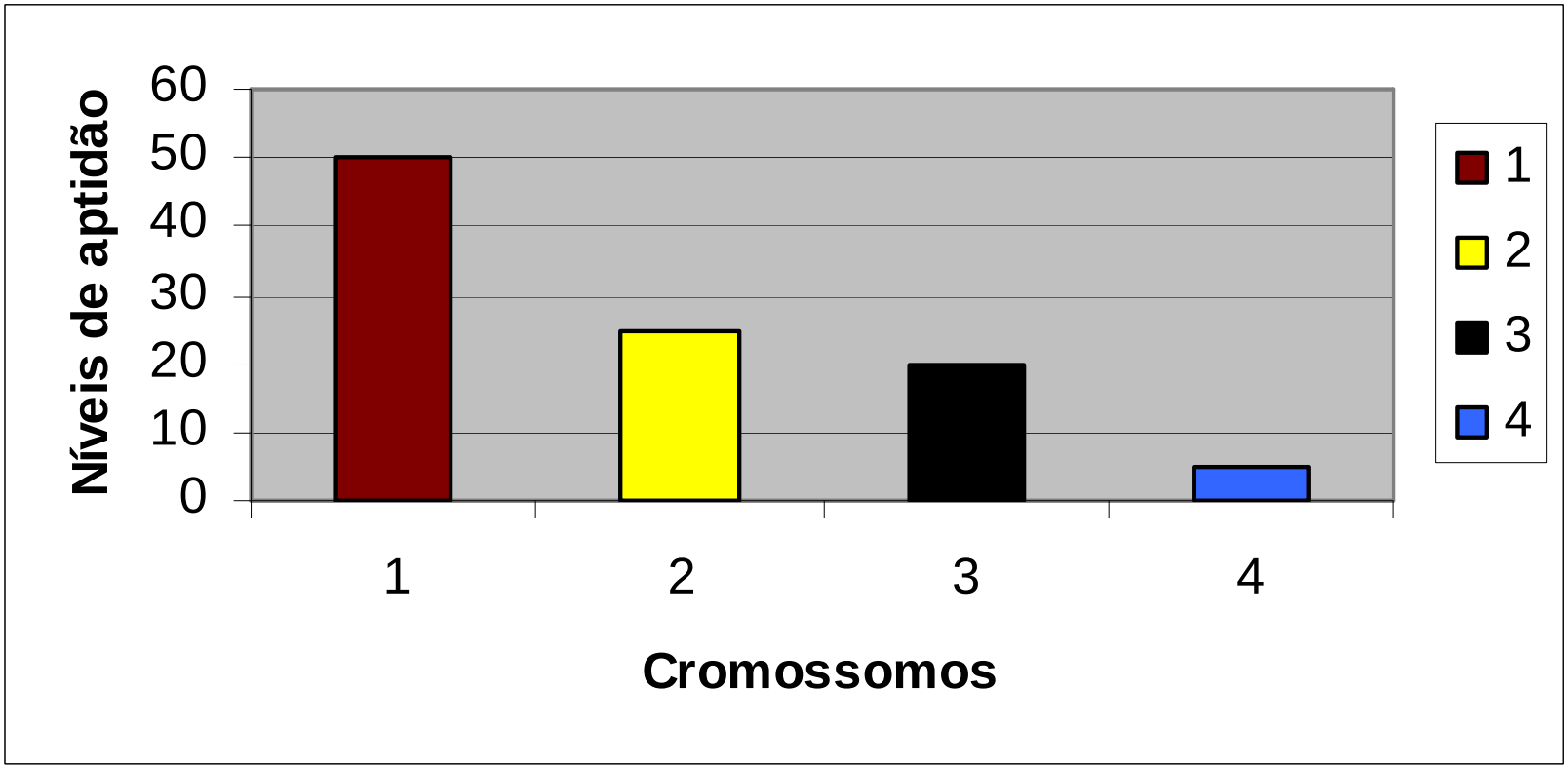
- O operador *crossover* pode gerar um ou mais filhos
- Operador **mutação** clássico



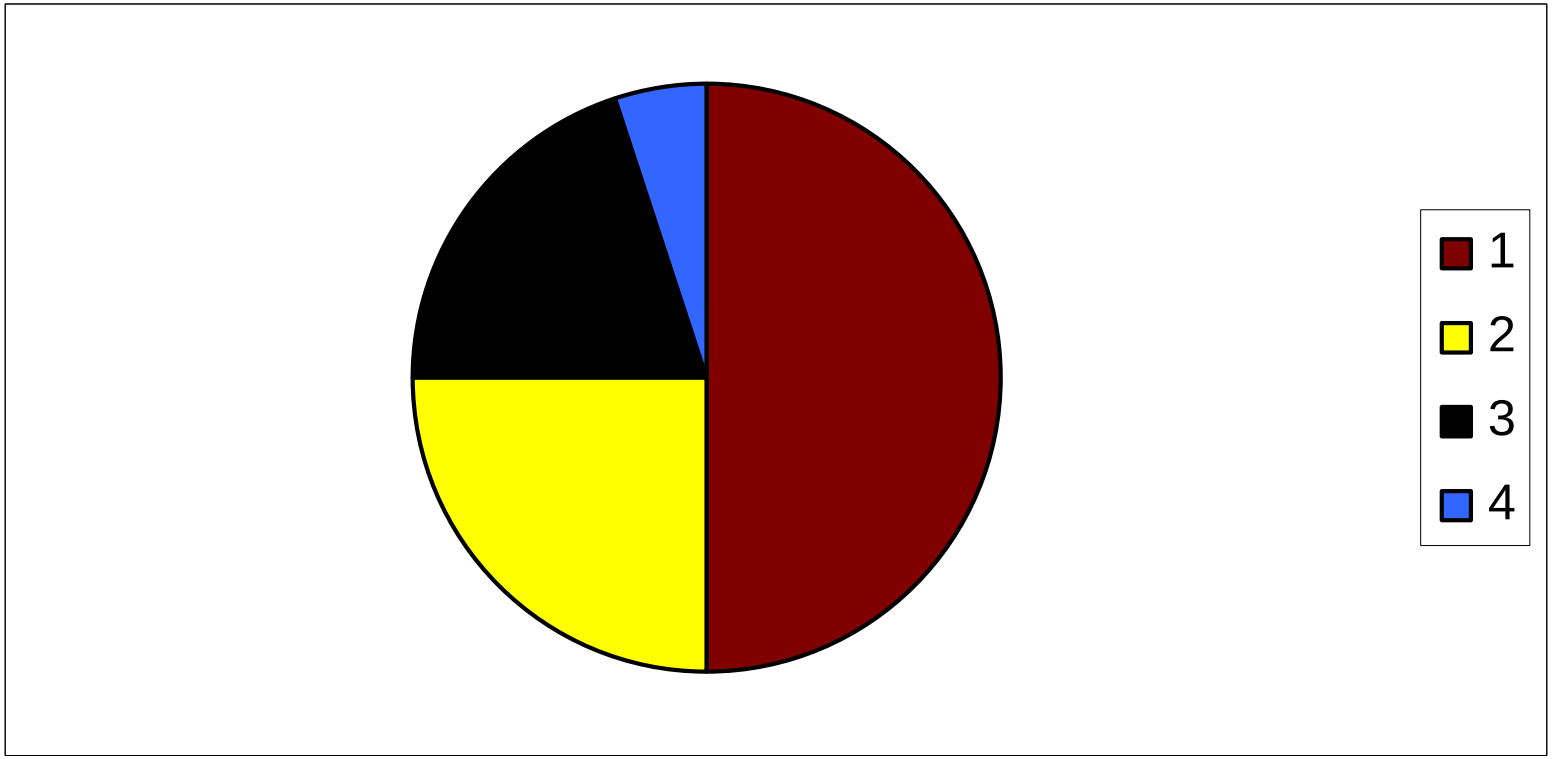
Sobrevivência / morte de cromossomos

- Como selecionamos os cromossomos que devem sobreviver?
- Sobrevivem os que possuem os melhores níveis de aptidão?
- É importante permitir também a sobrevida de cromossomos menos aptos, do contrário o método ficaria preso em ótimos locais
- Elitismo

Seleção de cromossomos sobreviventes



Roleta Russa (Seleção de sobreviventes)



Iterated Local Search (ILS)

- Método de busca local
- Explora o espaço de soluções por meio de perturbações nas soluções ótimas locais, seguidas de procedimentos de busca local
- Se ao aplicar a busca local não for encontrada uma solução melhor, a perturbação é aumentada
- Sempre que é encontrada uma solução melhorada, a perturbação volta ao nível mais mínimo
- Termina quando um critério de parada for atingido, p. ex., tempo de processamento

Iterated Local Search (ILS)