Tratamento de problemas $\mathcal{NP}$-difíceis: Heurísticas

- Heurísticas são algoritmos que geram soluções viáveis para quais *não se pode dar garantias de qualidade*. Ou seja, não se sabe o quão *distante* a solução gerada está de uma solução ótima (5% ?, 10% ?, 50% ?, 100% ?, ...).
- Tipos de heurísticas:
  - *construtivas*: normalmente adotam estratégias *gulosas* para construir as soluções. Tipicamente são aplicadas a problemas onde é fácil obter uma solução viável.
  - *de busca local*: partem de uma solução inicial e, através de transformações bem definidas, visitam outras soluções até atingir um *critério de parada* pré-definido.

Heurísticas Construtivas (TSP)

▷ Exemplo 1: TSP em um grafo não orientado completo.

```
Vizinho-Mais-Próximo(n, d)      (* d: matriz de distâncias *)
    Para i = 1 até n faça visitado[i] ← Falso ;
    visitado[1] ← Verdadeiro;
    ciclo ← {}, comp ← 0 e k ← 1;
    Para i = 1 até n − 1 faça
        j* ← argmin{d[k, j] : visitado[j] = Falso};
        visitado[j*] ← Verdadeiro ;
        ciclo ← ciclo ∪ {(k, j*)};      comp ← comp + d[k, j*];
        k ← j*;
    fim-para
    ciclo ← ciclo ∪ {(k, 1)};      comp ← comp + d[k, 1];
Retorne  comp.
```

▷ Complexidade: $O(n^2)$

Heurísticas Construtivas (TSP)

▷ Exemplo 2: heurística para o TSP ≡ algoritmo de Kruskal para AGM.

```
TSP-Guloso(n, d)      (* d: matriz de distâncias *)
    L ← lista das arestas ordenadas crecentemente pelo valor de d;
    Para i = 1 até n faça grau[i] ← 0;   componente[i] = i   fim-para
    k ← 0;      ciclo ← {};      comp ← 0;
    Enquanto k ≠ n faça
        (u, v) ← Remove-primeiro(L);
        Se (grau[u] ≤ 1 e grau[v] ≤ 1 e componente(u) ≠ componente(v))
        ou (grau[u] = grau[v] = 1 e k = n − 1) então
            ciclo ← ciclo ∪ {(u, v)};      comp ← comp + d[u, v];
            Unir-componentes(u, v);
            grau[u] + +;      grau[v] + +;      k + +;
        fim-se
    fim-enquanto
Retorne  comp.
```

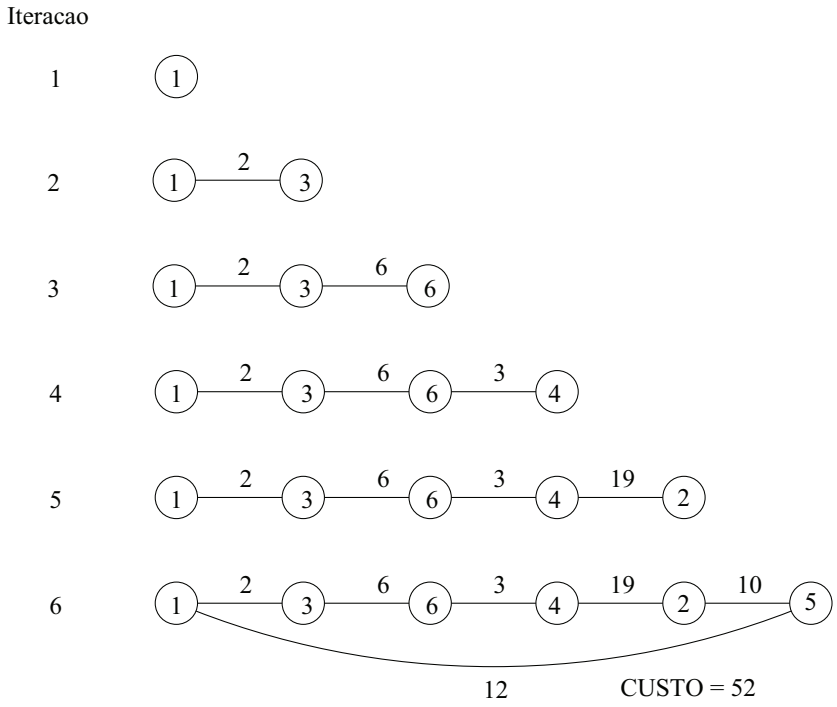
▷ Complexidade: $O(n^2 \log n)$ (usar *compressão de caminhos* para *união de conjuntos disjuntos*).

Heurísticas Construtivas (TSP)

Aplicação das heurísticas para o TSP:

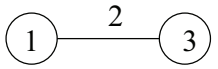
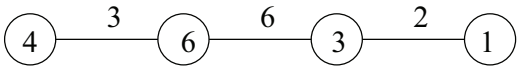
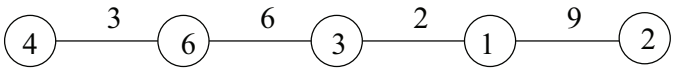
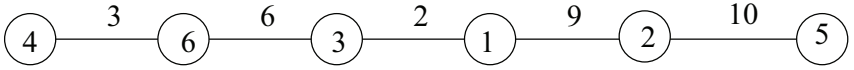
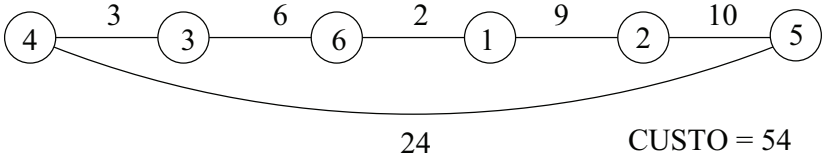
$$d = \begin{bmatrix} - & 9 & 2 & 8 & 12 & 11 \\ 9 & - & 7 & 19 & 10 & 32 \\ 2 & 7 & - & 29 & 18 & 6 \\ 8 & 19 & 29 & - & 24 & 3 \\ 12 & 10 & 18 & 24 & - & 19 \\ 11 & 32 & 6 & 3 & 19 & - \end{bmatrix}$$

Vizinho-Mais-Próximo



Heurísticas Construtivas (TSP)

TSP-GULOSO

Iteracao	
1	
2	
3	
4	Aresta (2,3) rejeitada (grau de 3)
5	Aresta (1,4) rejeitada (subciclo)
6	
7	
9, 10 , 11, 12	Rejeita as arestas (1,6), (1,5), (3,5), (2,4) e (5,6) (subciclos)
13	

Heurísticas Construtivas (Mochila)

▷ Exemplo 2: Problema da Mochila.

```
Mochila-guloso(c, w, W)
    Ordenar itens segundo a razão c_i / w_i;
    (* assuma que c_1/w_1 ≥ c_2/w_2 ≥ ... ≥ c_n/w_n *)

    W̄ ← W;      S ← {};
    Para i = 1 até n faça
        Se w_i ≤ W̄ então
            W̄ ← W̄ − w_i;
            S ← S ∪ {i};
        fim-se
    fim-para
Retorne S.
```

▷ Complexidade: $O(n \log n)$.

Heurísticas Construtivas (Mochila)

- Aplicação da heurística `Mochila-guloso`.

$$\begin{array}{ll} \text{maximize} & 8x_1 + 16x_2 + 20x_3 + 12x_4 + 6x_5 + 10x_6 + 4x_7 \\ \text{Sujeito a} & 3x_1 + 7x_2 + 9x_3 + 6x_4 + 3x_5 + 5x_6 + 2x_7 \leq 17, \\ & x \in \mathbb{B}^7. \end{array}$$

Observação: $\frac{8}{3} \geq \frac{16}{7} \geq \frac{20}{9} \geq \frac{12}{6} \geq \frac{6}{3} \geq \frac{10}{5} \geq \frac{4}{2}$

- Solução gulosa: $S = \{1, 2, 4\}$, custo $= 36$.
- Solução ótima: $S = \{1, 2, 6, 7\}$, custo $= 38$.

Heurísticas Construtivas

- Soluções gulosas podem ser *arbitrariamente ruins* !
- `Mochila-guloso` é arbitrariamente ruim.
- Instância: $W = n$, $c_1 = 3/n$, $w_1 = 2/n$ e, para todo $i = 2, \ldots, n$, $c_i = n - (1/n)$ e $w_i = n - (1/n)$.
  Observação: $\frac{c_1}{w_1} \geq \frac{c_2}{w_2} = \ldots = \frac{c_n}{w_n}$.
- Solução gulosa: $S = \{1\}$, custo $= 3/n$.
- Solução ótima: $S = \{2\}$, custo $= n - (1/n)$.
- $\lim_{n\to\infty} \frac{(3/n)}{n-(1/n)} = 0$.
  Ou seja, aumentando o valor de n nesta instância, a solução gulosa pode se afastar tanto quanto eu quiser da solução ótima !

Heurísticas Construtivas

- ▷ `Vizinho-Mais-Próximo` para o TSP é arbitrariamente ruim.
- ▷ Instância: matriz simétrica de distâncias d onde, para $i < j$, tem-se:

$$d[i,j] = \begin{cases} n^2, & \text{se } i = n-1 \text{ e } j = n, \\ 1, & \text{se } j = i+1, \\ 2, & \text{caso contrário.} \end{cases}$$

- ▷ Solução gulosa: ciclo $= \{1, 2, \ldots, n-1, n\}$ e comp $= n^2 + n$.
- ▷ Solução ótima: ciclo $= \{1, 2, \ldots, n-3, n, n-2, n-1\}$ e comp $= n + 3$.
- ▷ $\lim_{n\to\infty} \frac{n+3}{n^2+n} = 0$.

  Novamente, aumentando o valor de n nesta instância, a solução gulosa pode se afastar tanto quanto eu quiser da solução ótima !

Heurística de Inserção Mais Barata

TSP-INSERÇÃO-MAIS-BARATA (G, V, E, c), G é completo

1 Seja C um circuito qualquer (e.g. fecho convexo).

2 Enquanto C não é hamiltoniano

3 Seja $(u, v) \in C$ e $x \notin C$ tal que $c(u, x) + c(x, v) - c(u, v)$ é mínimo

4 $C \leftarrow C - (u, v) + (u, x) + (x, v)$.

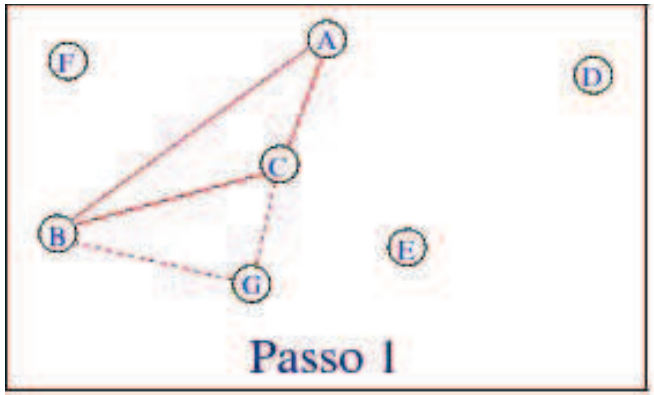
5 devolva C

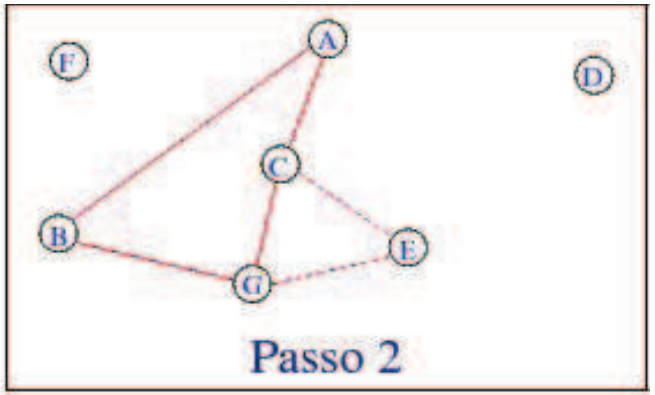
Por ser uma heurística útil e fácil de implementar, foi analisada com mais detalhes.

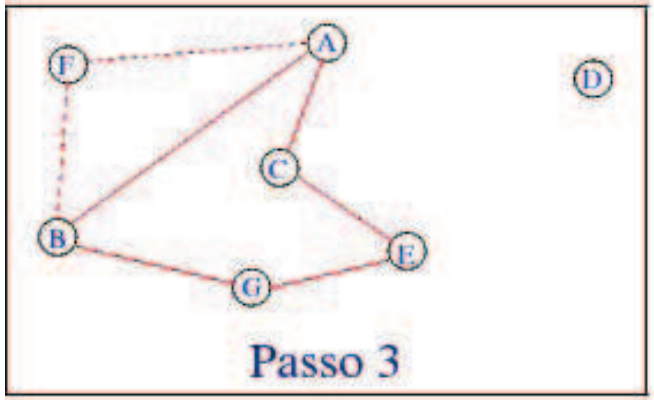
Teorema: *Rosenkrantz, Stearns, Lewis: TSP-Vizinho-Mais-Próximo é uma 2-aproximação para grafos métricos.*

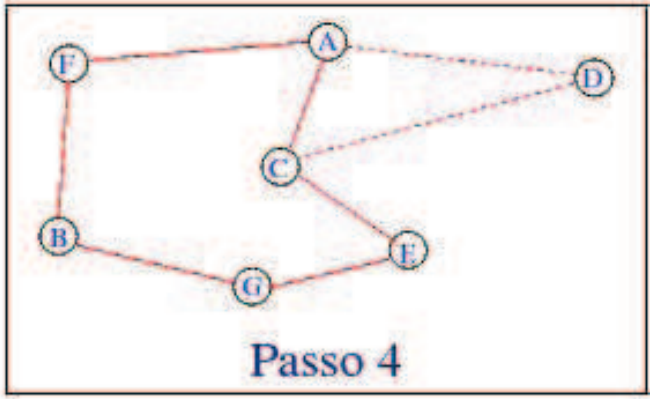
Prova. Exercício. □

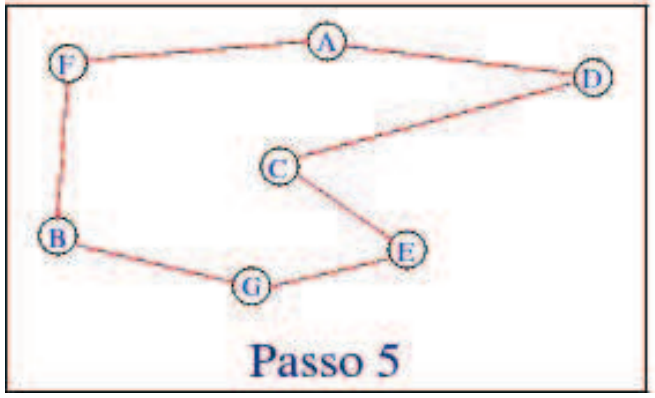
Simulação do TSP-Inserção-mais-barata











Exemplo de solução obtida pelo TSP-Inserção-mais-barata

Heurística de Inserção do Mais Distante

TSP-INSERÇÃO-MAIS-DISTANTE (G, V, E, c), G é completo

1 Seja C um circuito qualquer (e.g. fecho convexo).
2 Enquanto C não é hamiltoniano
3 Denote por $c(v, C)$ a menor distância de v a um vértice de C.
4 Seja $x \notin C$ um vértice tal que $c(x, C)$ é máxima.
5 Insira x em C na posição onde há menor aumento de custo
6 devolva C

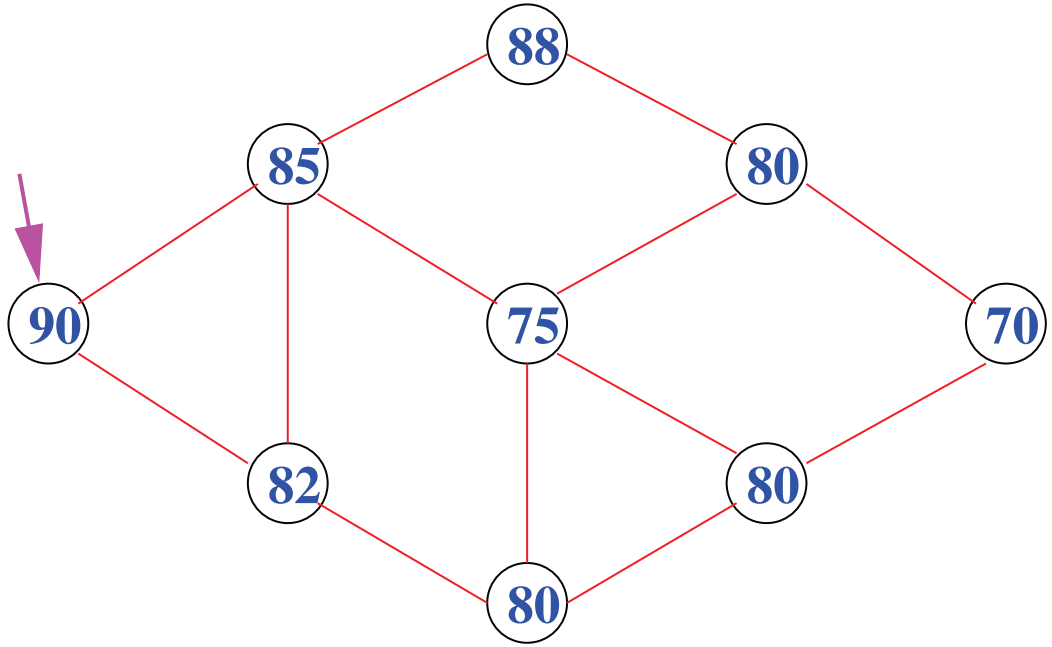
Esta heurística dá resultados melhores na prática que o TSP-Inserção-mais-barata. Mas não há prova de sua aproximação. O pior caso são fatores de 6.5 em uma métrica e de 2.43 no plano euclidiano [Hurkens'92].

Busca Local

- Começam com uma atribuição ou solução
- Iterativamente fazem operações locais melhorando a solução anterior
- Quando não puder melhorar, devolvem a solução obtida

Grafo de Vizinhanças

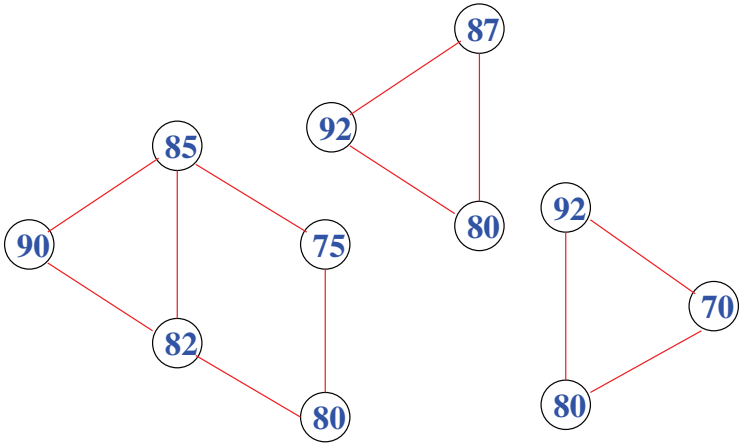
- Vértices são “soluções” viáveis: em geral é um conjunto muito grande
- Arestas indicam transformações de uma solução para outra
- Exemplo de grafo de vizinhança e uma solução inicial (e seu valor)



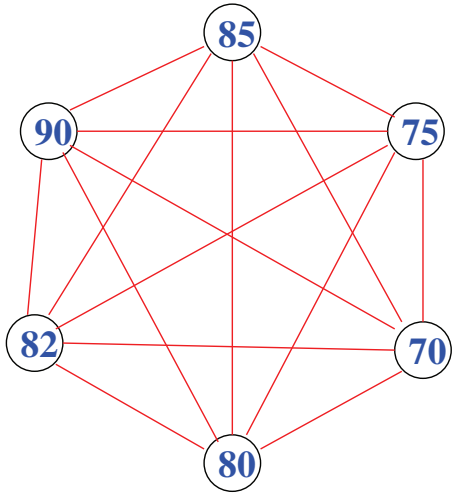
- Objetivo: Chegar na solução de melhor custo pelo grafo de vizinhanças

Grafos de vizinhança ruins

- Grafo desconexo:Podemos nunca chegar na solução ótima



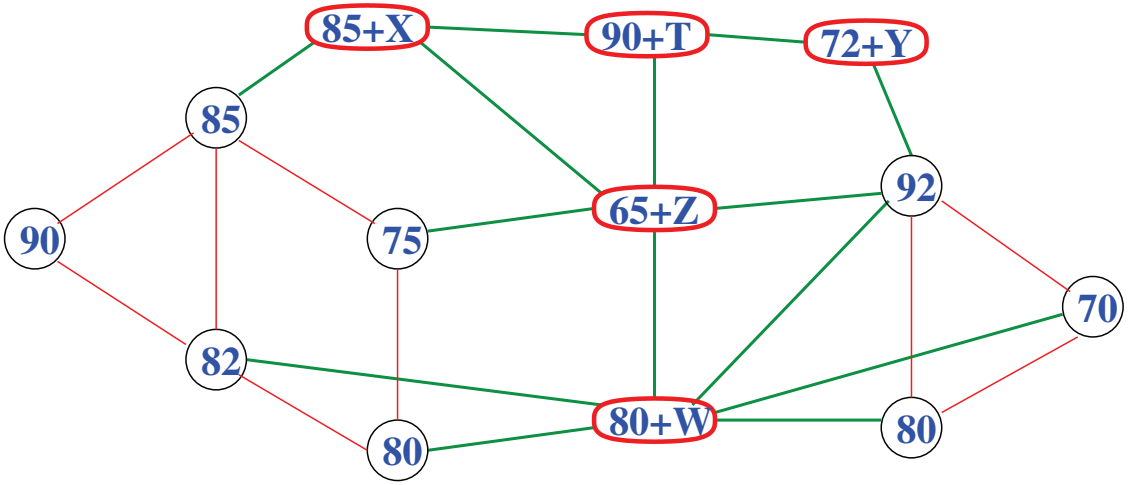
- Grafo denso: Percorrer os vizinhos
  de uma “solução” tem a mesma complexidade do problema original



Aumentando a conectividade do grafo de vizinhança

Idéia: Inserir nós inviáveis e considerar um custo adicional conforme o grau de inviabilidade

- Aumenta possibilidades de sair de uma solução para outras
- Possibilidades de sair de mínimos locais por soluções inviávies
- Soluções iniciais podem ser inviáveis



- Seja $\mathcal{S}$ conjunto das soluções viáveis do problema
- $c(S)$ é o custo de S + $p(S)$, onde $p(S)$ é uma penalidade de acordo com o 'grau de inviabilidade' de S

Heurísticas de Busca Local e Hill Climbing

- Uso de vizinhança entre soluções viáveis
- Uso de solução inicial
- Melhorias sucessivas a partir da solução atual

Notação:

I = Instância do problema

$\mathcal{N}$ = Conjunto de soluções viáveis para I

$\mathcal{N}(S)$ = Conjunto de soluções vizinhas a S (no grafo de vizinhanças)

$c(S)$ = valor da solução S

Hill Climbing

VIZINHO-MELHOR (S, I)

1 se existe vizinho $S' \in \mathcal{N}(S)$ com valor melhor que S
2 então retorne S'
3 senão retorne $\emptyset$

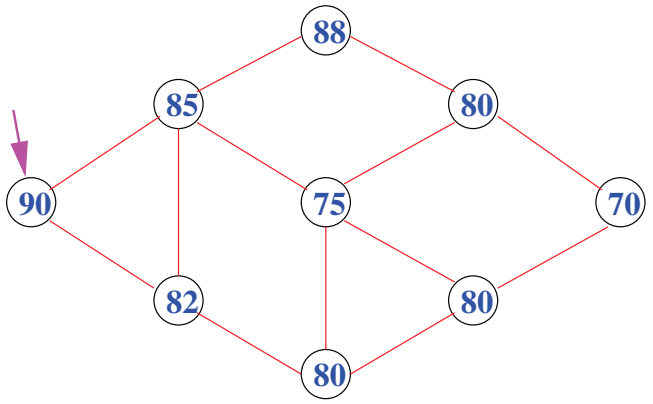
BUSCA-LOCAL-GERAL (I)

1 encontre "solução" inicial $S \in \mathcal{N}$ para I
2 $S' \leftarrow$VIZINHO-MELHOR(S, I)
3 enquanto $S' \neq \emptyset$ faça
4 $S \leftarrow S'$
5 $S' \leftarrow$VIZINHO-MELHOR(S, I)
6 devolva S

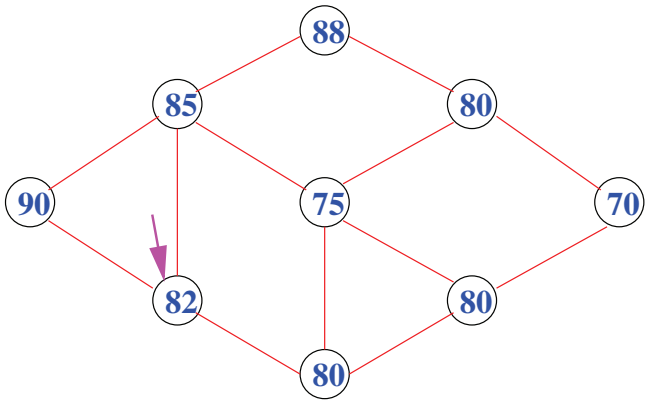
Busca Local Geral (minimização):

Escolher o melhor dentre todos os vizinhos que tem valor melhor

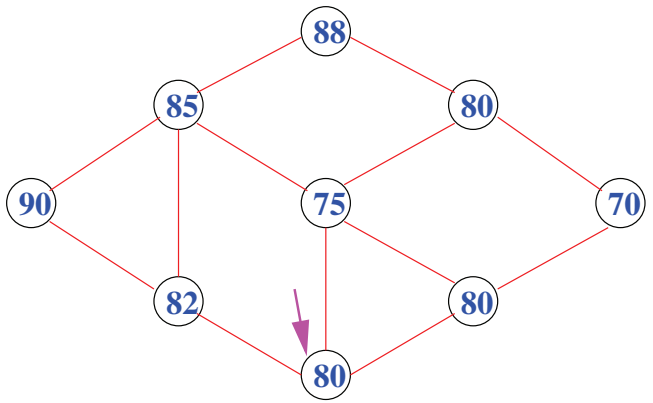
Grafo de vizinhança entre "soluções" viáveis



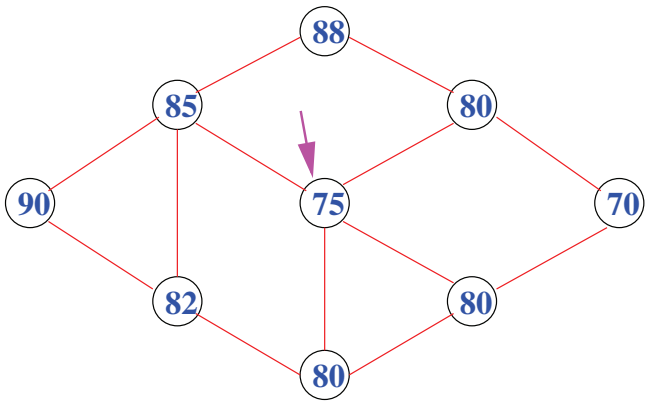
Solução inicial de valor 90



Solução de valor 82

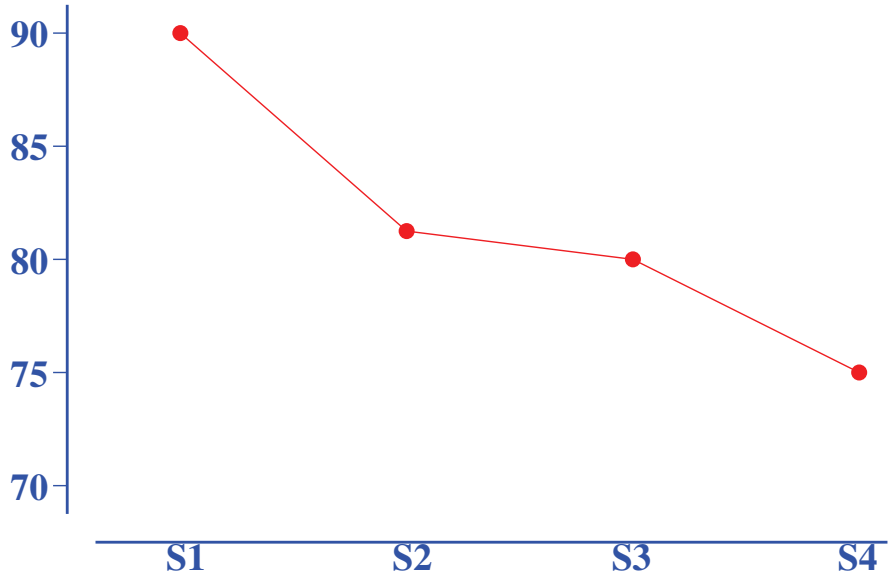


Solução de valor 80

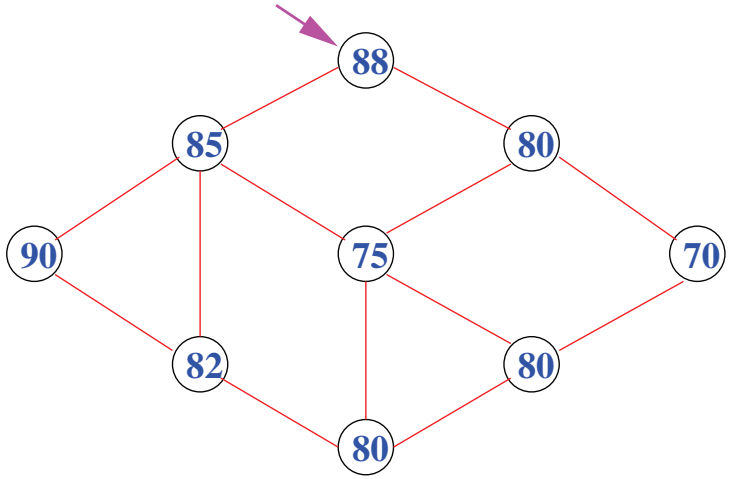


Solução ótima local de valor 75

Comportamento do algoritmo de busca local

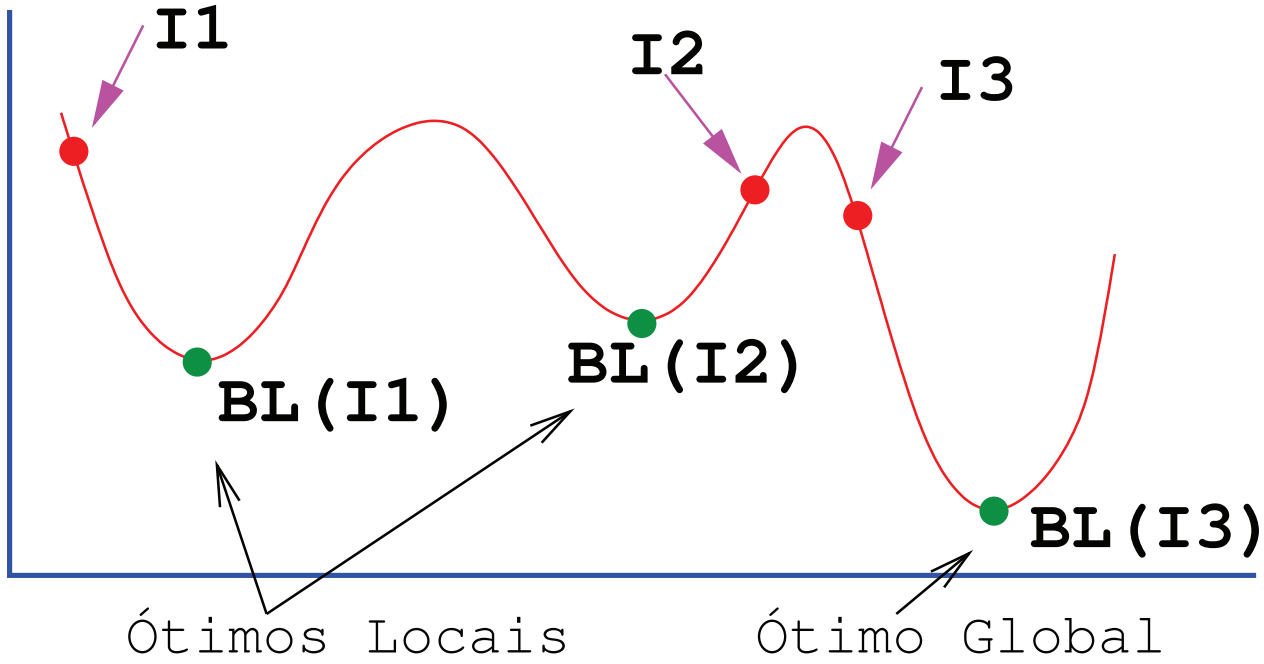


Se a primeira solução fosse a de valor 88, teríamos chegado na solução ótima

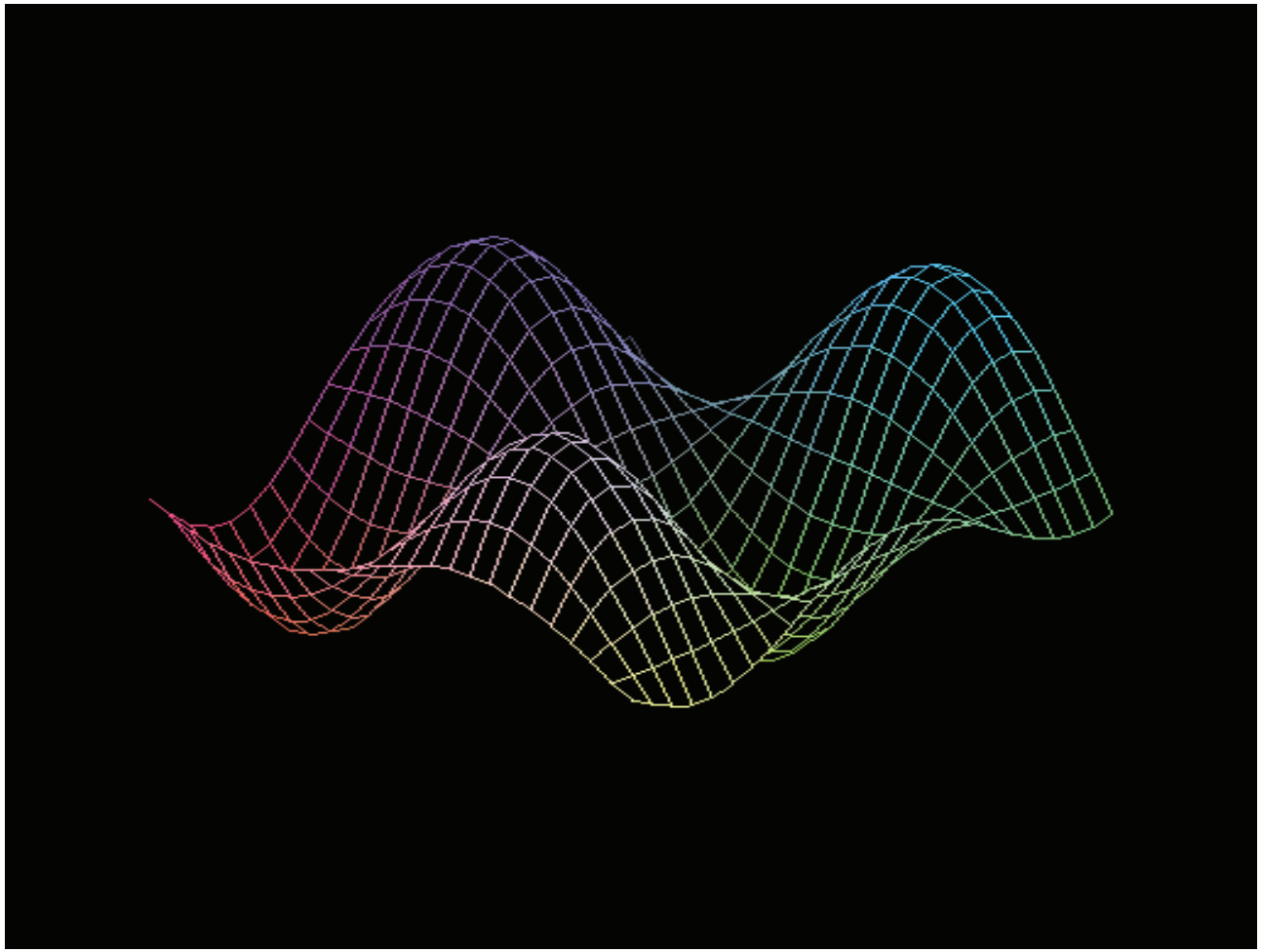


Saindo de mínimos locais: Multi-Start Local Search

- **Executar algoritmo de Busca Local com diferentes inícios**
- **Guardar melhor solução**
- **Exemplo para minimização (unidimensional)**



Mínimos e máximos locais em função bidimensional



Busca Local para o TSP

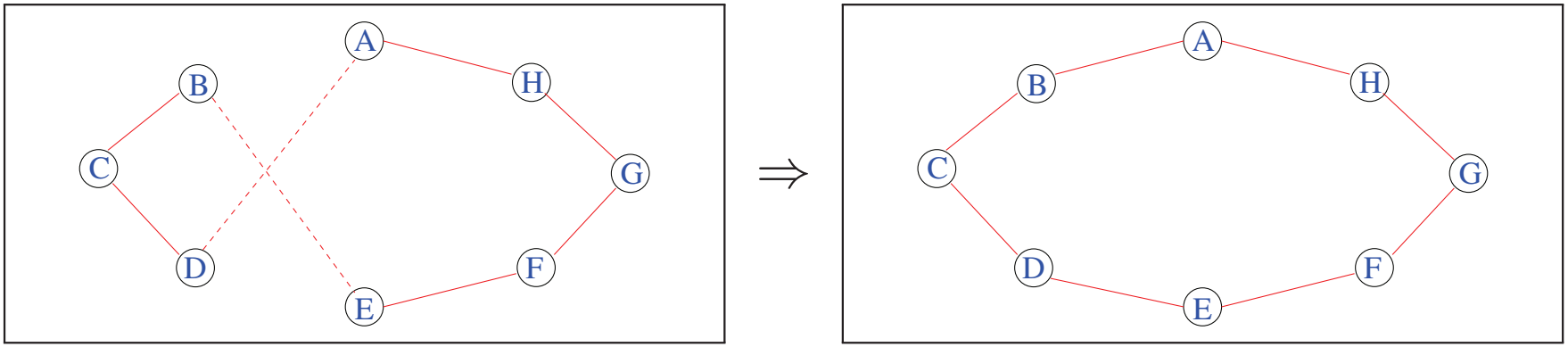
Considere grafos completos

Vizinhança-K-OPT(C) := $\{C' : C'$ é circuito hamiltoniano obtido de C removendo K arestas e inserindo outras K arestas.$\}$.

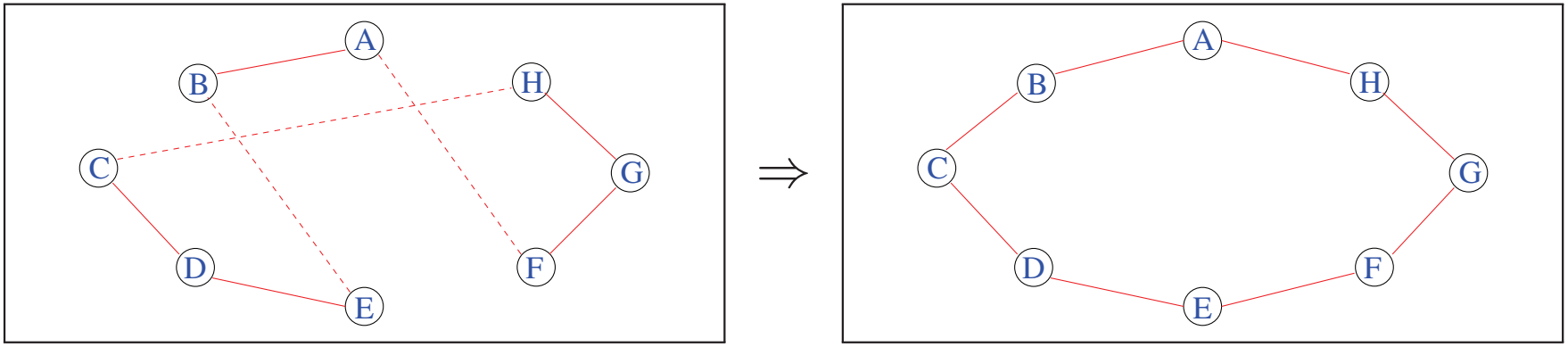
K-OPT $(G = (V, E, c))$

1 encontre um circuito hamiltoniano inicial C
2 repita
3 procure C' em Vizinhança-K-OPT(C) tal que $\mathrm{val}(C') < \mathrm{val}(C)$.
4 se encontrou tal C', $C \leftarrow C'$
5 até não conseguir encontrar tal C' no passo 3
6 devolva C

Exemplo de troca 2-OPT



Exemplo de troca 3-OPT



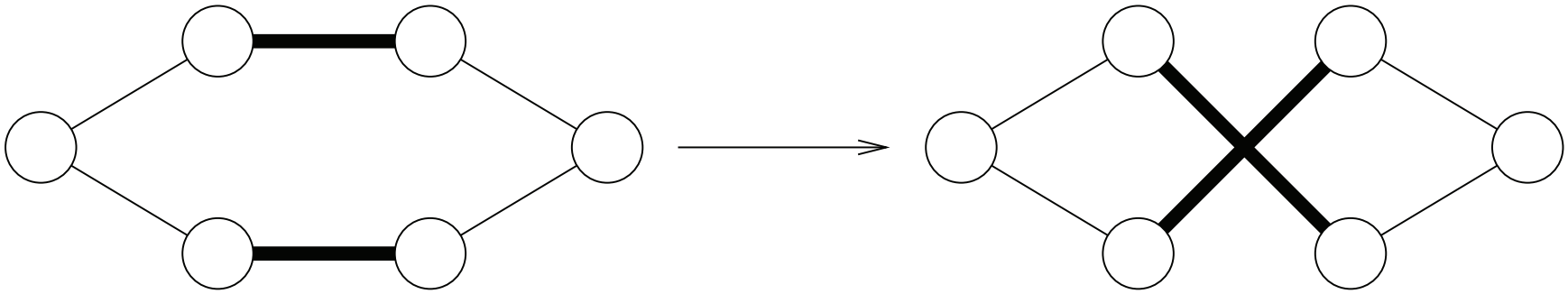
Uma solução viável pode ter vários vizinhos usando troca 3-OPT.
Quantos ?

Heurísticas de Busca Local

- Exemplo 2:
  A tupla é um vetor representando uma permutação de $\{1, \ldots, n\}$.
  $N_2(t)$: conjunto de todas as tuplas obtidas trocando-se as posições de dois elementos da permutação.
  Complexidade: $\Theta(n^2)$.
- Algoritmo de busca local (problema de minimização):
  - Encontrar uma solução inicial t.
  - Encontrar t' em $N(t)$ com menor custo.
  - Se o custo de t' é menor que o custo de t, fazer $t \leftarrow t'$ e repetir o passo anterior. Se não, retorne t e pare.

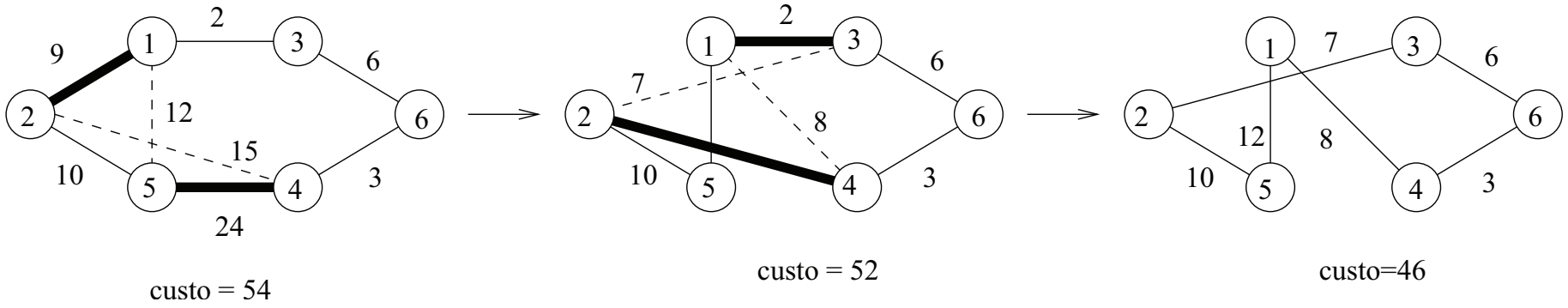
Heurísticas de Busca Local (TSP)

- ▷ Heurística da 2-troca para o TSP (Lin e Kernigham).
- ▷ Ciclo representado por uma permutação dos n vértices.
- ▷ Vizinhança: substituir pares de arestas.



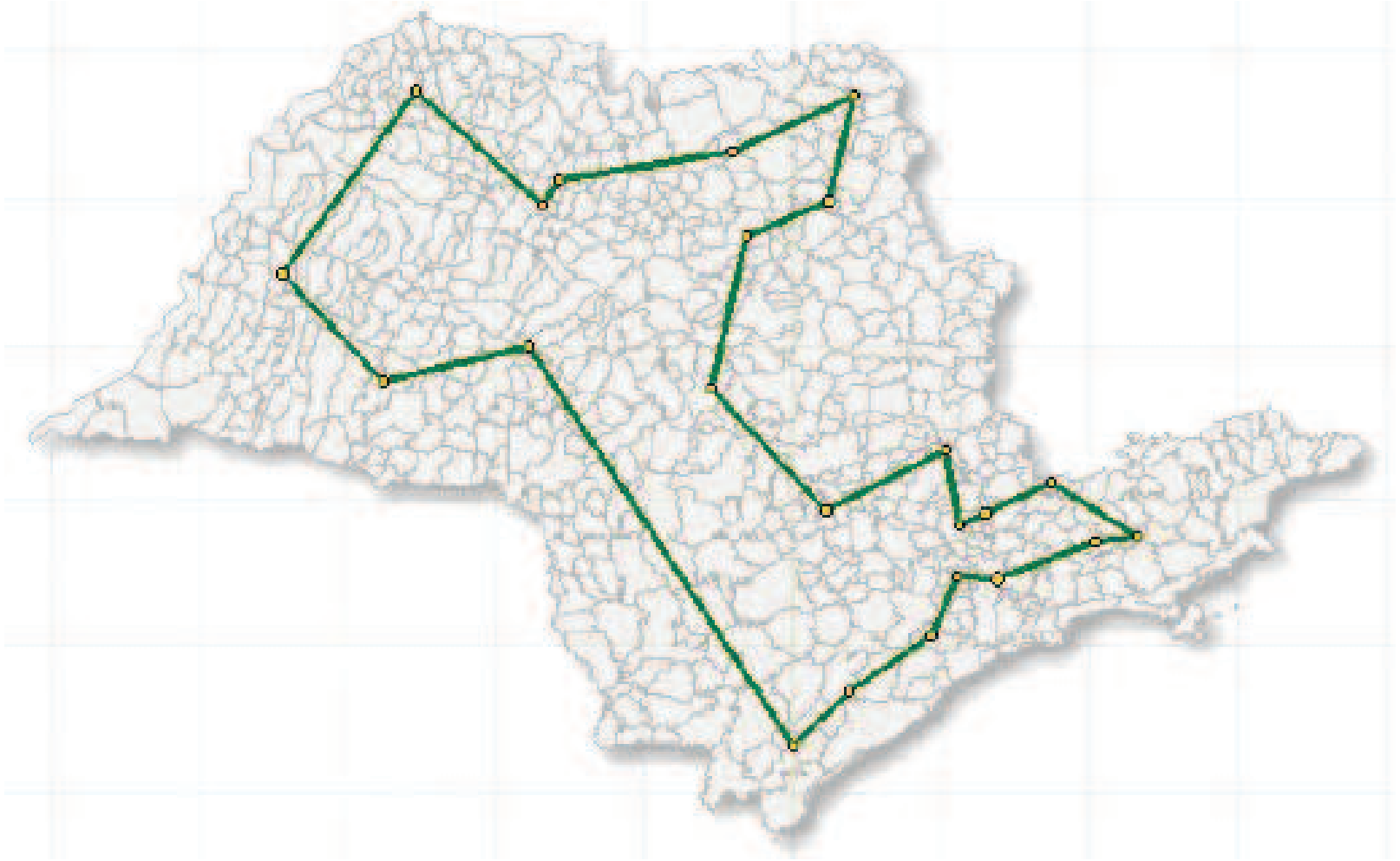
- ▷ Complexidade: $\Theta(n^2)$.

Heurísticas de Busca Local (TSP)



- ▷ Tuplas: vetor de permutações de 1 até n.
- ▷ Vizinhança:
  inverte seqüência entre posições i e $j \pmod{n}$ $(j \geq i+2)$.
- ▷ No exemplo:
  $(1,3,6,\underline{4},5,2,\underline{1}) \Longrightarrow (\underline{1},3,6,4,\underline{2},5,1) \Longrightarrow (1,4,6,3,2,5,1)$

Exemplo de solução obtida pelo TSP-2-OPT



Comparação em grafos euclidianos aleatórios

D.S. Johnson & L.A. McGeoch'97

- **Circuito inicial por algoritmo guloso (estilo Kruskal)**
  - Inserindo arestas mais leves primeiro
  - Descartando arestas que inviabilizam solução
- **Comparando com limitante inferior do ótimo**
  - Limitante de Held-Karp

$N =$	10^2	$10^{2.5}$	10^3	$10^{3.5}$	10^4	$10^{4.5}$	10^5	$10^{5.5}$	10^6
Guloso	19.5	18.8	17.0	16.8	16.6	14.7	14.9	14.5	14.2
2-OPT	4.5	4.8	4.9	4.9	5.0	4.8	4.9	4.8	4.9
3-OPT	2.5	2.5	3.1	3.0	3.0	2.9	3.0	2.9	3.0

Fator de excesso em relação ao limitante de Held-Karp

Heurísticas de Busca Local (Partição de Grafos)

- ▷ Entrada: grafo não orientado $G = (V, E)$, com $|V| = 2n$, e custos c_{ij} para toda aresta $(i, j) \in E$.
- ▷ Saída: um subconjunto $V' \subseteq V$, com $|V'| = n$ e que minimize o valor de $\sum_{i \in V'} \sum j \notin V' c_{ij}$.
- ▷ Solução representada por um vetor a de $2n$ com os valores de 1 até $2n$. Nas n primeiras posições estão os vértices de V' e nas n seguintes os vértices de $\overline{V'}$.
- ▷ Vizinhança: todas as trocas possíveis de pares de vértices $(a[i], a[j])$, onde $1 \leq i \leq n$ e $(n+1) \leq j \leq 2n$.
- ▷ Complexidade: $\Theta(n^2)$.

Heurísticas de Busca Local (Partição de Grafos)

▷ Exemplo: grafo completo com 6 vértices (K_6).

$$c = \begin{bmatrix} - & 9 & 2 & 8 & 12 & 11 \\ 9 & - & 7 & 19 & 10 & 32 \\ 2 & 7 & - & 29 & 18 & 6 \\ 8 & 19 & 29 & - & 24 & 3 \\ 12 & 10 & 18 & 24 & - & 19 \\ 11 & 32 & 6 & 3 & 19 & - \end{bmatrix}$$

Solução inicial:
$a = \{1, 4, 6, 2, 3, 5\}$.

- vizinhos $(1,2)$ $(1,3)$ $(1,5)$ $(4,2)$ $(4,3)$ $(4,5)$ $(6,2)$ $(6,3)$ $(6,5)$
  ganho -29 -12 -7 $\underline{-66}$ -15 -40 -22 -43 -32

- Nova solução: $a = \{1, 2, 6, 4, 3, 5\}$.

- vizinhos $(1,4)$ $(1,3)$ $(1,5)$ $(2,4)$ $(2,3)$ $(2,5)$ $(6,4)$ $(6,3)$ $(6,5)$
  ganho 37 34 23 66 51 26 44 59 54

- *Ótimo Local !*

Heurísticas de Busca Local

- ▷ Pode ser vantajoso que a busca local passe por soluções inviáveis !
- ▷ Nesses casos a função objetivo é composta de duas parcelas:

$$g(.) = f(.) + \alpha h(.),$$

onde f é função original, h é uma função que mede quão inviável é a solução e α é um fator de penalização.

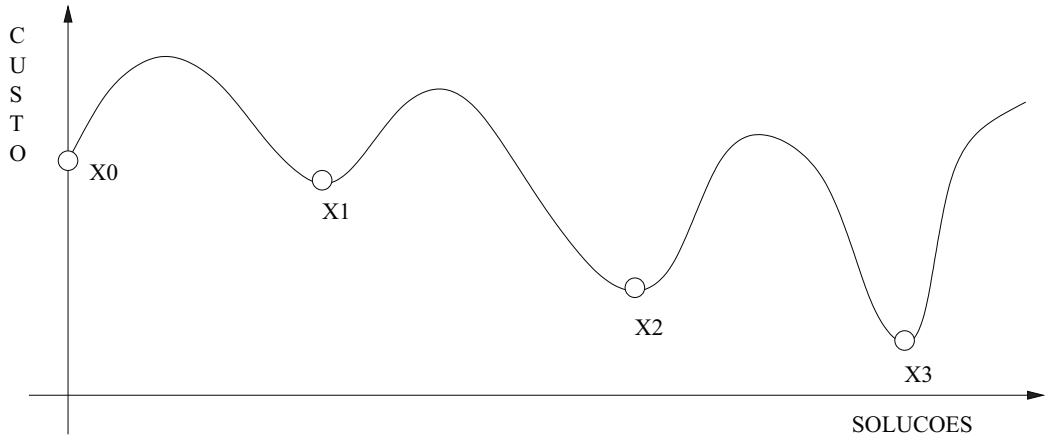
- ▷ Exemplo: no problema da partição de grafos, considere a vizinhança onde só um vértice muda de V' para $\overline{V'}$ ou vice-versa.
- ▷ Penalizar as soluções inviáveis usando $\alpha > 0$ grande e definindo:

$$h(V', \overline{V'}) = ||V'| - |\overline{V'}||^2.$$

- ▷ Se acabar em uma solução inviável, aplicar um algoritmo guloso que rapidamente restaura a viabilidade.

Heurísticas de Busca Local

- Busca local retorna solução que é ótimo local.



- Escapando de ótimos locais: mover para melhor vizinho mesmo se o custo for pior.
- *Metaeurísticas*: Busca Tabu, Simulated Annealing, Algoritmos genéticos,etc.