Universidade do Minho
Escola de Engenharia
Departamento de Produção e Sistemas

Problema do caixeiro viajante

Filipe Alvelos
falvelos@dps.uminho.pt

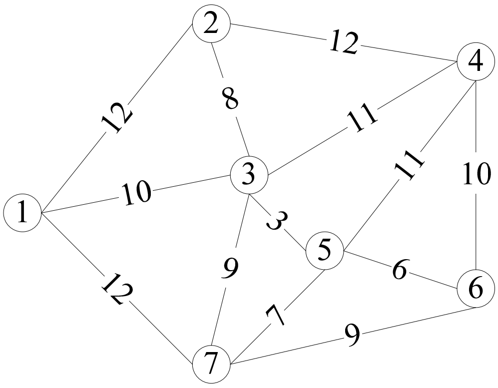
2020

Índice

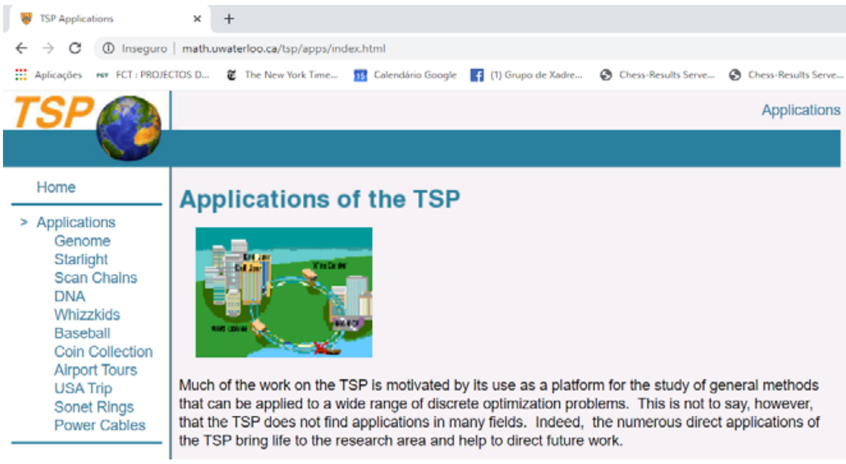
- Introdução
- Heurísticas construtivas
  - Vizinho mais próximo (pura, multi-início e aleatorizada)
  - Aresta de menor custo
  - Árvore de suporte
  - Inserção
- Heurísticas de pesquisa local
  - Troca de duas arestas (2-opt)
  - Troca de três arestas (3-opt)
  - Inserção de cidade

Introdução

- Visitar um conjunto 7 cidades regressando àquela de que se partiu, percorrendo a menor distância possível. As distâncias são dadas na figura. Cada cidade tem de ser visitada uma e uma só vez.



Introdução

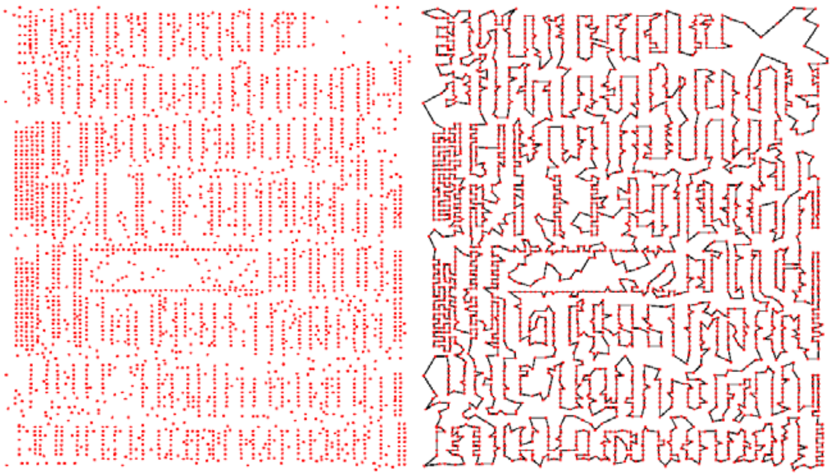


Introdução

- Número de alternativas para uma instância do n cidades é $n!$ (todas as permutações das n cidades)
- Para $n=10$, número de possibilidades: $10! = 3628800$
- Para $n=30$, número de possibilidades: $30! = 2.65 \times 10^{32}$
  - Testando um bilião de alternativas por segundo (um computador muito bom!), o tempo total seria mais de oito milénios!
  - Estima-se que a idade do universo seja 4.4×10^{17} segundos
- Para $n=100$, número de possibilidades: $100! = 9.3 \times 10^{157}$
  - Estima-se que o número de átomos no universo esteja entre 10^{78} e 10^{82}

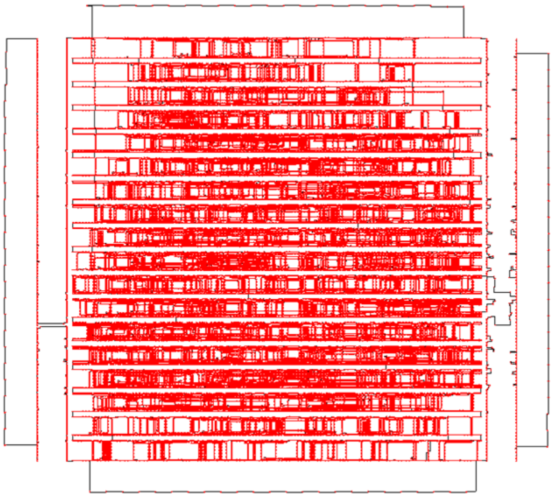
Introdução

- Instância com 3038 cidades e uma solução óptima



Introdução

- Maior instância do TSP resolvida até à optimalidade (85900 cidades)



Introdução

- Prémio para quem conseguir arranjar um algoritmo exacto eficiente para o TSP: 1 milhão de dólares (ver http://www.claymath.org/millennium-problems, o TSP é NP).
- Mona Lisa challenge (resolver instância com 100 000 cidades) http://www.math.uwaterloo.ca/tsp/data/ml/monalisa.html

Introdução

- n número de cidades
- c_{ij} custo (distância) em que se incorre ao atravessar o arco $ij, \forall ij \in A$
- Problema do caixeiro viajante simétrico (STSP): $c_{ij} = c_{ji}$
- Problema do caixeiro viajante assimétrico (ATSP): $\exists ij \in A: c_{ij} \neq c_{ji}$
- Assume-se que existe um arco entre qualquer par de cidades
- Caso não exista pode considerar-se um arco artificial com um custo suficientemente elevado

Introdução

- Modelo de programação inteira de Miller-Tucker-Zemlin
- Solução é definida como um conjunto de arcos (variáveis x)
- Variáveis auxiliares (u) evitam sub-circuitos definindo uma sequência com início na cidade 1

$$x_{ij} = \begin{cases} 1, se\ arco\ ij\ faz\ parte\ do\ circuito \\ 0, caso\ contrário \end{cases}, \forall ij \in A$$

$$u_i - posição\ em\ que\ é\ visitada\ a\ cidade\ i, i = 2, \dots, n$$

$$Min\ z = \sum_{ij \in A} c_{ij} x_{ij}$$

$$sujeito\ a:$$

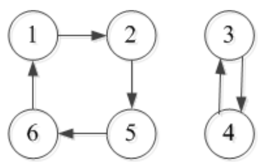
$$\sum_{j:ij \in A} x_{ij} = 1, \forall i \in N$$

$$\sum_{j:ji \in A} x_{ji} = 1, \forall i \in N$$

$$u_j \geq n(x_{ij} - 1) + u_i, \forall ij \in A, i \neq 1, j \neq 1$$

$$x_{ij} \in \{0,1\}, \forall ij \in A$$

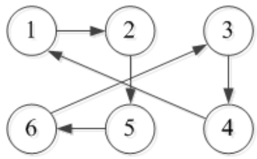
$$u_i \geq 0, i = 2, \dots, n$$



Programação inteira

- Restrição $u_j \geq n(x_{ij}-1) + u_i$ para arco ij
  - Para $x_{ij} = 1$, reduz-se a $u_j \geq u_i + 1$, o que implica que o nodo j é visitado depois do nodo i
  - Para $x_{ij} = 0$, é equivalente a $u_j \geq u_i + 1 - n$, que é redundante
    - como $n \geq u_i + 1$, a restrição $u_j \geq u_i + 1 - n$ é equivalente a $u_j \geq 0$
- Exemplo de solução

$$x_{12} = x_{25} = x_{56} = x_{63} = x_{34} = x_{41} = 1$$
$$u_2 = 0, u_3 = 3, u_4 = 4, u_5 = 1, u_6 = 2$$

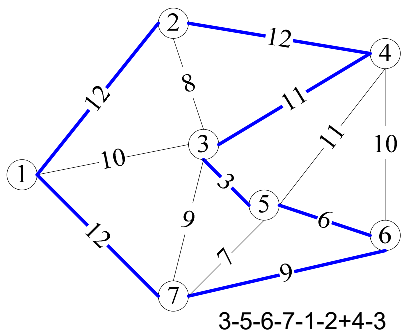


Heurísticas construtivas

- Heurística do vizinho mais próximo
- Considerar um nodo arbitrariamente e um caminho formado apenas por esse nodo
- Enquanto há nodos que não fazem parte do caminho
  - Determinar o nodo mais próximo em relação ao último considerado (de entre os que não fazem parte do caminho)
  - Incluí-lo no caminho
- Formar o circuito, incluindo no caminho a aresta que liga o último nodo considerado ao primeiro

Heurísticas construtivas

- i1: Solução obtida partindo de 3, Custo 65

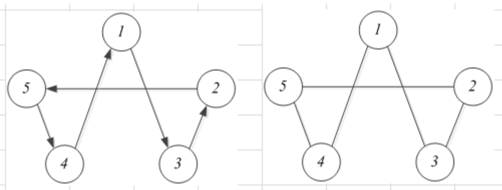


3-5-6-7-1-2+4-3
Comprimento total=
3+6+9+12+12+12+11

- i2: Solução obtida partindo de 1

	1	2	3	4	5
1	-	6	3	7	11
2	6	-	7	10	9
3	3	7	-	8	7
4	7	10	8	-	4
5	11	9	7	4	-

Custo=30



começando em 1-3-5-6-7 - impossível
Comprimento total=infinito

Heurísticas construtivas

- Exemplo – Heurística do vizinho mais próximo (início em 1)

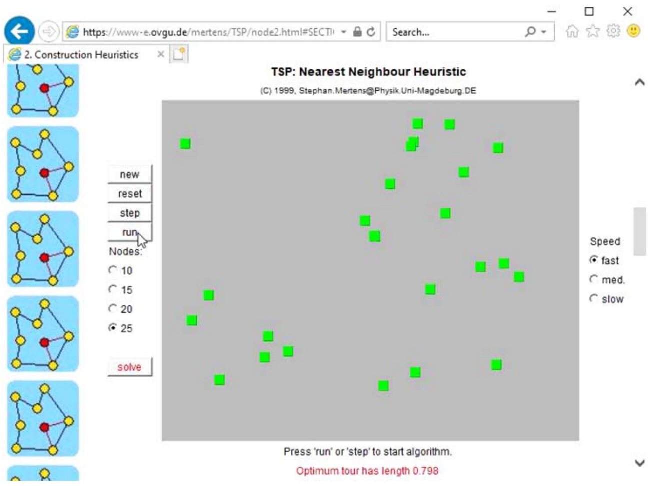
	1	2	3	4	5	6
1	100	2	6	4	15	11
2	2	100	4	19	12	2
3	6	4	100	7	8	9
4	4	19	7	100	17	4
5	15	12	8	17	100	1
6	11	2	9	4	1	100

	1	2	3	4	5	6
1	100	2	6	4	15	11
2	2	100	4	19	12	2
3	6	4	100	7	8	9
4	4	19	7	100	17	4
5	15	12	8	17	100	1
6	11	2	9	4	1	100

- Solução dada pela heurística do vizinho mais próximo:1-2-6-5-3-4-1, valor 24

Heurísticas construtivas

- Exemplo – Heurística do vizinho mais próximo (distâncias são as distâncias euclideanas)



Heurísticas construtivas

- Heurística do vizinho mais próximo multi-início
- Repetir heurística do vizinho mais próximo com início em diferentes nodos

	1	2	3	4	5	6
1	100	2	6	4	15	11
2	2	100	4	19	12	2
3	6	4	100	7	8	9
4	4	19	7	100	17	4
5	15	12	8	17	100	1
6	11	2	9	4	1	100

- Soluções temporárias
  - Início em 1: 1-2-6-5-3-4-1, valor 24
  - Início em 2: 2-1-4-6-5-3-2, valor 23
  - Início em 3: 3-2-1-4-6-5-3, valor 23
  - Início em 4: 4-1-2-6-5-3-4, valor 24
  - Início em 5: 5-6-2-1-4-3-5, valor 24
  - Início em 6: 6-5-3-2-1-4-6, valor 23
- Solução dada pela heurística do vizinho mais próximo multi-início: 2-1-4-6-5-3-2, valor 23

Heurísticas construtivas

- Heurística do vizinho mais próximo aleatorizada
- Conceito de lista restrita de candidatos (RCL – restricted candidate list): em cada iteração não é necessariamente seleccionado o nodo mais próximo mas um escolhido aleatoriamente de entre os que pertencem à RCL
- Um vizinho j faz parte da RCL se o seu custo não estiver acima de uma determinada proporção da amplitude entre o maior e menor custo (minimização):

$$c_j \leq c_{min} + \alpha(c^{max} - c_{min})$$

- Exemplo com no início no nodo 1 para $\alpha = 0.5$
  - $c_{min} = 2, c_{max} = 15$
  - $c_j \leq 2 + 0.5(15 - 2) \equiv c_j \leq 8.5$
  - $RCL = \{2,3,4\}$

	1	2	3	4	5	6
1	100	2	6	4	15	11
2	2	100	4	19	12	2
3	6	4	100	7	8	9
4	4	19	7	100	17	4
5	15	12	8	17	100	1
6	11	2	9	4	1	100

Heurísticas construtivas

- Algoritmo
  - Definir valor da constante α
  - Começar num nodo arbitrário
  - Repetir até todas os nodos fazerem parte do circuito
    - Definir os nodos da RCL
    - Seleccionar aleatoriamente (probabilidade uniforme) um nodo da RCL
    - Adicionar o nodo seleccionado à solução
- Elementos com o melhor valor nunca são excluídos
- A proporção é dada pelo parâmetro α, $\alpha \in [0,1]$, que controla o nível de aleatoriedade e de “greediness”
- $\alpha = 0$, é equivalente à heurística do vizinho mais próximo
- $\alpha = 1$, é equivalente a escolher uma solução aleatoriamente

Heurísticas construtivas

- Exemplo com no início no nodo 1 para $\alpha = 0.5$
  - $c_{min} = 2, c_{max} = 15$
  - $c_j \leq 2 + 0.5(15 - 2) \equiv c_j \leq 8.5$
  - $RCL = \{2,3,4\}$, logo é seleccionado 2, 3, ou 4 com igual probabilidade
  - Supondo que é seleccionado o 4
- Solução actual 1-4
  - $c_{min} = 4, c_{max} = 19$
  - $c_j \leq 4 + 0.5(19 - 4) \equiv c_j \leq 12.5$
  - $RCL = \{3,6\}$
  - Supondo que é seleccionado o 3
- Solução actual 1-4-3
  - $c_{min} = 4, c_{max} = 9$
  - $c_j \leq 6.5, RCL = \{2\}$
- …

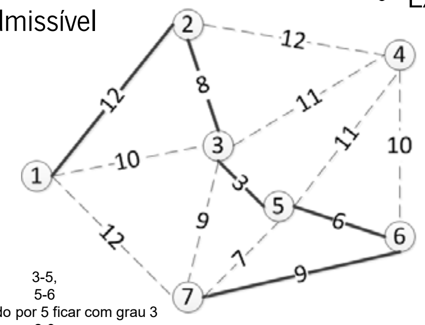
	1	2	3	4	5	6
1	100	2	6	4	15	11
2	2	100	4	19	12	2
3	6	4	100	7	8	9
4	4	19	7	100	17	4
5	15	12	8	17	100	1
6	11	2	9	4	1	100

Heurísticas construtivas

- Heurística da aresta com menor custo
- Uma aresta é *não* admissível se
  - faz parte do conjunto de arestas já seleccionadas para o circuito
  - a sua selecção implica que um nodo fica com grau *3*
  - a sua selecção implicar a existência de um circuito com menos de n arestas
- Algoritmo
  - De todas as arestas admissíveis seleccionar aquela que tem menor custo
  - Repetir enquanto as arestas seleccionadas não formarem um circuito com n arestas

Heurísticas construtivas

- Exemplo 1: Solução não admissível

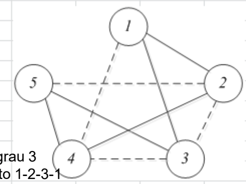


3-5,
5-6
5-7 excluído por 5 ficar com grau 3
2-3
3-7 excluído por 3 ficar com grau 3
6-7
1-3 excluído por 3 ficar com grau 3
4-6 excluído por 6 ficar com grau 3
3-4 excluído por 3 ficar com grau 3
4-5 excluído por 5 ficar com grau 3
1-2
1-7 excluído por formar subcircuito 1-2-3-5-6-7-1
2-4 excluído por 2 ficar com grau 3
Comprimento total=
~~3+6+8+9+12+12~~ infinito

- Exemplo 2: Solução

	2	3	4	5
1	6	3	7	11
2		7	10	9
3			8	7
4				4

Custo=30

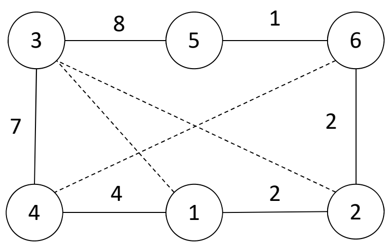


1-3
4-5
1-2
1-4 excluído por 1 ficar com grau 3
2-3 excluído por formar subcircuito 1-2-3-1
3-5
3-4 excluído (grau 3 e subcircuito)
2-5 excluído (grau 3)
2-4
Comprimento total=6+3+10+7+4

Heurísticas construtivas

- Exemplo – Heurística da aresta de menor custo

	1	2	3	4	5	6
1	100	2	6	4	15	11
2	2	100	4	19	12	2
3	6	4	100	7	8	9
4	4	19	7	100	17	4
5	15	12	8	17	100	1
6	11	2	9	4	1	100



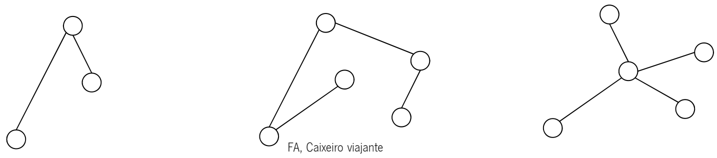
- Solução dada pela heurística da aresta de menor custo:1-2-6-5-3-4-1, valor 24

Heurísticas construtivas

- Circuito Euleriano (de Euler (1707-1783)) é um circuito que inclui todos os arcos uma e uma só vez (nodos podem ser repetidos)
- Teorema de Euler: um grafo não orientado tem um circuito Euleriano se e só se todos os vértices têm grau par
- Ideia de heurísticas baseadas em árvores de suporte de custo mínimo é gerar um circuito Euleriano com base na árvore de suporte de custo mínimo e transformá-lo num circuito Hamiltoniano
- (Um circuito Hamiltoniano é um circuito que passa uma e uma só vez por cada vértice (i.e. uma solução do TSP))

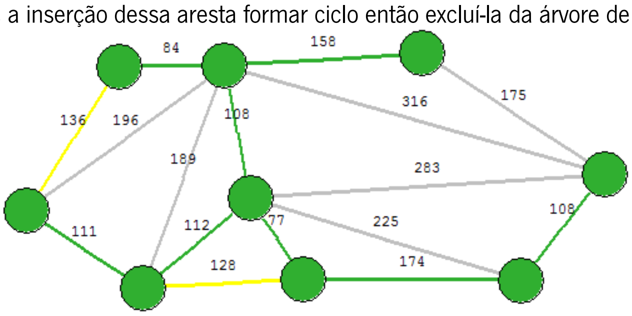
Heurísticas construtivas

- Heurística da árvore de suporte de custo mínimo
  - Primeiro passo: obter árvore de suporte de custo mínimo (conjunto de arcos de baixo custo que não formam ciclos – embora vértices possam ter grau maior que 2)
  - Segundo passo: transformar árvore de suporte em caixeiro viajante
- Um grafo não orientado com n vértices é uma árvore de suporte se tem $n-1$ arestas e não tem circuitos
- Uma árvore de suporte garante a conectividade da rede (existe um caminho entre qualquer par de vértices) com o menor número de arestas possível



Heurísticas construtivas

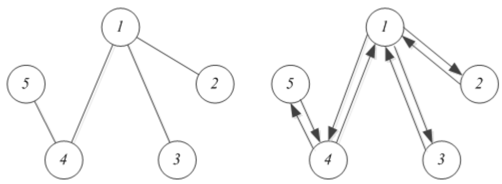
- Existem algoritmos eficientes para obter a árvore de suporte com menor custo como algoritmo de Kruskal
- Enquanto não estiverem seleccionadas $n-1$ arestas
  - Considerar a aresta de menor custo das que ainda não foram examinadas
  - Se a inserção dessa aresta não formar um ciclo então incluí-la na árvore de suporte
  - Se a inserção dessa aresta formar ciclo então excluí-la da árvore de suporte



Heurísticas construtivas

1. Obter árvore de suporte de custo mínimo
2. Duplicar todos os arcos da árvore de suporte de custo mínimo
3. Obter um circuito em que todos os arcos são percorridos uma e uma só vez (circuito Euleriano)
4. Transformar o circuito Euleriano num circuito em que todos os nodos são percorridos uma e uma só vez (circuito Hamiltoniano)

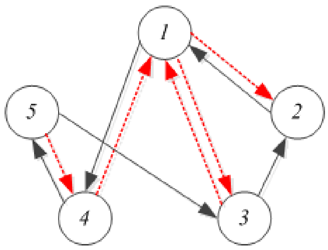
	2	3	4	5
1	6	3	7	11
2		7	10	9
3			8	7
4				4

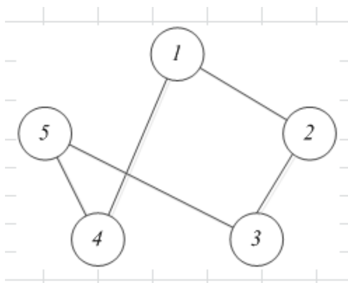


Circuito Euleriano 1-4-5-4-1-3-1-2-1
Circuito Hamiltoniano 1-4-5-3-2-1

Heurísticas construtivas

- Circuito Euleriano 1-4-5-4-1-3-1-2-1
- Circuito Hamiltoniano 1-4-5-3-2-1

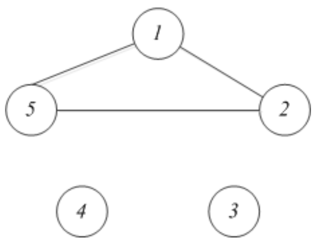




Heurísticas construtivas

- Heurísticas de inserção

1. Construir um circuito com duas cidades
2. De acordo com um critério pré-estabelecido, seleccionar um vértice k que não faz parte do circuito e uma aresta ij que faz parte do circuito
3. Adicionar k ao circuito entre i e j
4. Repetir 2 e 3 até o circuito incluir todos os vértices

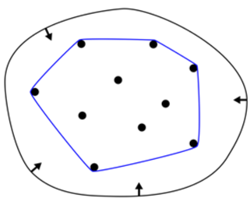


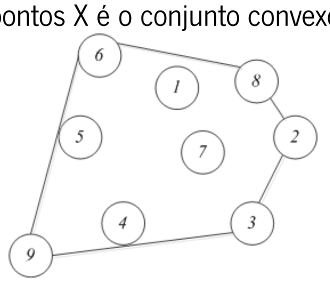
Heurísticas construtivas

- Heurística de inserção do vértice mais próximo
  - Para todos os vértices que não fazem parte do circuito, calcular a sua distância ao circuito (a menor distância entre ele e um vértice do circuito)
  - Seleccionar o vértice k que tem menor distância ao circuito
  - Seleccionar o arco ij do circuito que tem menor valor de $c_{ik} + c_{kj} - c_{ij}$
- Heurística de inserção do vértice mais afastado
  - Para todos os vértices que não fazem parte do circuito, calcular a sua distância ao circuito (a menor distância entre ele e um vértice do circuito)
  - Seleccionar o vértice k que tem maior distância ao circuito
  - Seleccionar o arco ij do circuito que tem menor valor de $c_{ik} + c_{kj} - c_{ij}$
- Heurística da inserção mais barata
  - De todas as combinações entre arestas do circuito (ij) e vértices (k) que não fazem parte do circuito, seleccionar a que tem o menor valor de $c_{ik} + c_{kj} - c_{ij}$

Heurísticas construtivas

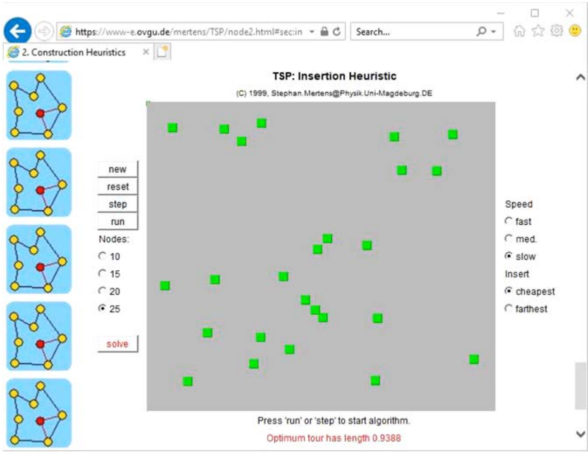
- Em instâncias euclideanas, há vantagem em usar para circuito (parcial) inicial o invólucro convexo
- TSP Euclideanas
  - As cidades estão todas no mesmo plano
  - As distâncias entre as cidades são dadas pela distância Euclideana, i.e., $c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ onde (x_i, y_i) e (x_j, y_j) são as coordenadas da cidade i e da cidade j
- Invólucro convexo de um conjunto de pontos X é o conjunto convexo mais pequeno que contém X





Heurísticas construtivas

- Exemplo – Heurística da inserção mais barata partindo do invólucro convexo (distâncias são as distâncias euclideanas)

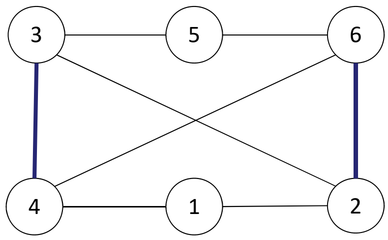


Pesquisa Local

- Perspectiva das heurísticas de pesquisa local não é a de construir uma solução mas a de modificar uma solução existente para (potencialmente) obter uma solução melhor
- Em geral, não há forma óbvia de identificar as modificações que melhoram a solução, mas se forem em número reduzido, pode-se experimentar todas
- Algoritmo de pesquisa local
  - Obter uma solução actual (possivelmente com base numa heurística construtiva)
  - Obter a melhor solução vizinha da solução actual
  - Se a melhor vizinha é melhor do que a solução actual, então a melhor vizinha passa a ser a solução actual, ir para o passo anterior

Pesquisa Local

- Estrutura de vizinhança baseada em trocas de duas arestas (2-Opt)
- Exemplo solução actual 3-5-6-4-1-2-3



- Solução vizinha: 3-5-6-2-1-4-3
- A vizinhança de uma solução é o conjunto de soluções que se obtém ao remover duas arestas não adjacentes do circuito e a reconstruir o circuito com outras duas arestas
- Equivalente a inverter um caminho do circuito (um segmento da permutação): no exemplo, 3-5-6-**4-1-2**-3 passou a 3-5-6-**2-1-4**-3

Pesquisa Local

	1	2	3	4	5
1	-	6	3	13	11
2	6	-	7	10	9
3	3	7	-	8	15
4	13	10	8	-	4
5	11	9	15	4	-

- Heurística de pesquisa local 2-opt
- Uma vizinha 2-opt pode ser obtida invertendo uma subsequência (de comprimento até de n−2 cidades) da permutação associada à solução
- s é uma permutação
- Obter uma permutação, s^0, por aplicação da heurística do vizinho mais próximo

  $s^0 = (1,3,2,5,4)$
- Obter e avaliar todas as soluções que se obtêm a partir de s^0 através de uma operação de troca de duas arestas $\mathrm{N}(s^0) = \{t^{0,1}, t^{0,2}, t^{0,3}, t^{0,4}, t^{0,5}\}$

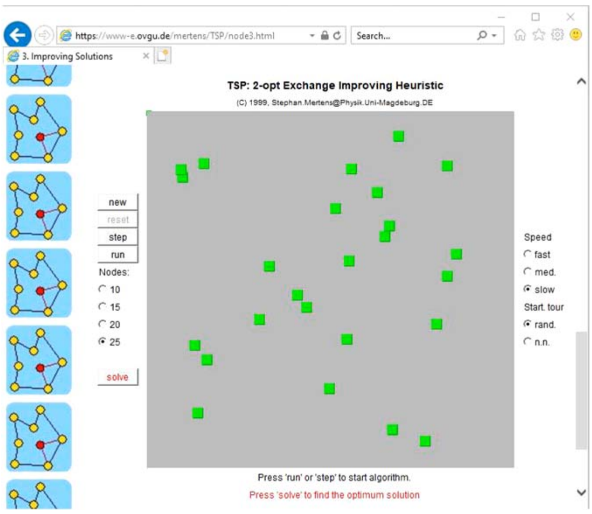
						Valor
s^0	1	3	2	5	4	36
$t^{0,1}$	1	2	3	5	4	45
$t^{0,2}$	1	5	2	3	4	48
$t^{0,3}$	1	3	5	2	4	50
$t^{0,4}$	1	3	4	5	2	30
$t^{0,5}$	1	3	2	4	5	35

Pesquisa Local

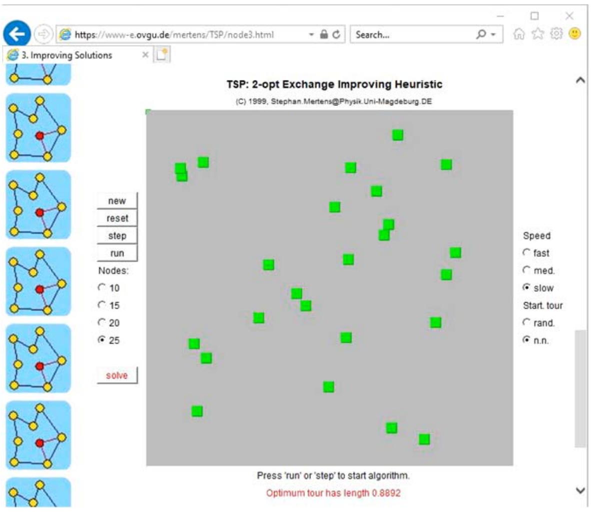
- Melhor solução obtida com troca de duas arestas é
  $t^{0,5} = (1,3,4,5,2)$
- Nova iteração
  $s^1 = t^{0,5} = (1,3,4,5,2)$
- Todas as soluções que se obtêm por troca de duas arestas são piores (ou iguais) do que a actual, logo a solução actual é uma solução óptima ***local*** (relativamente à estrutura de vizinhança de troca de duas arestas)

						Valor
s^1	1	3	4	5	2	30
$t^{1,1}$	1	4	3	5	2	51
$t^{1,2}$	1	5	4	3	2	36
$t^{1,3}$	1	3	5	4	2	38
$t^{1,4}$	1	3	2	5	4	36
$t^{1,5}$	1	3	4	2	5	41

Pesquisa local

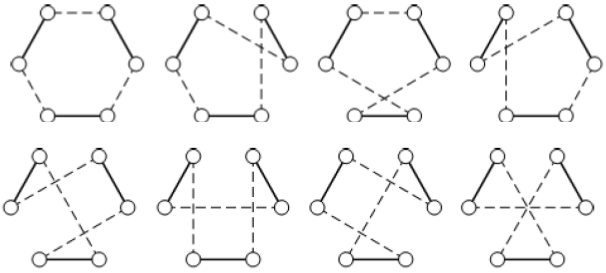


Pesquisa local



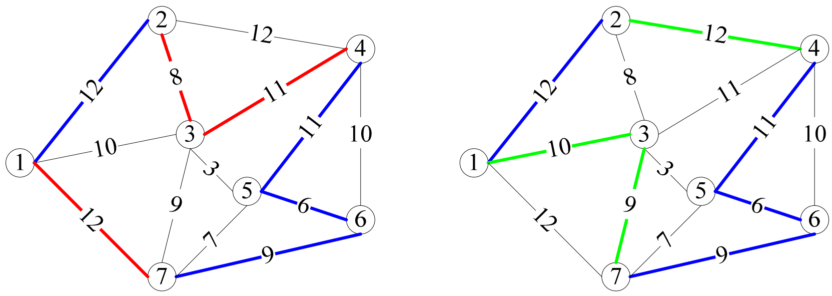
Pesquisa Local

- Estrutura de vizinhança baseada em trocas de três arestas (3-Opt)
- A vizinhança de uma solução é o conjunto de soluções que se obtém ao remover três arestas não adjacentes do circuito e a reconstruir o circuito com outras três arestas (pelo menos uma diferente)
- 7 soluções vizinhas (4 soluções com três novas arestas e 3 soluções com duas novas arestas)



Pesquisa Local

- Estrutura de vizinhança baseada em inserção de um vértice
- A vizinhança de uma solução é o conjunto de soluções que se obtém ao remover um vértice do circuito e inseri-lo numa posição diferente
- Exemplo de uma solução e uma sua vizinha



Considerações finais sobre heurísticas

- Outras heurísticas mais elaboradas que usam conceitos pouco gerais (mas específicos para o TSP)
  - Or-opt
  - Lin-Kernighan
  - GENI e GENIUS
- Heurísticas construtivas obtêm soluções a cerca de 10-15% do valor óptimo [JM]
- Heurísticas 3-opt obtém soluções a cerca de 3-4% do valor óptimo [JM]
- Heurística de Lin-Kernighan obtém soluções a cerca de 1-2% do valor óptimo [JM]
- ...nunca esquecer que existe um compromisso entre o tempo computacional e a qualidade das soluções obtidas

Metaheurísticas

- Ideia central das metaheurísticas: não parar a pesquisa num óptimo local
- GRASP (Greedy Randomized Adaptive Search Procedure)
  - Heurística construtiva aleatorizada + pesquisa local + multi-início
- VND (variable neighborhood descent) e VNS (variable neighborhood search)
  - Usam diferentes estruturas de vizinhança para iniciar pesquisas locais em soluções próximas de óptimos locais
- Pesquisa tabu
  - Aceitar a melhor solução vizinha (mesmo que seja pior do que a actual) permite continuar a pesquisa (tendo cuidado para não entrar em ciclo)
- Pesquisa por arrefecimento simulado (simulated annealing)
  - Solução vizinha é aceita com probabilidade que depende do seu valor e de pârametros que se vão alterando ao longo da execução do algoritmo (probabildiade de aceitação diminui com o tempo)