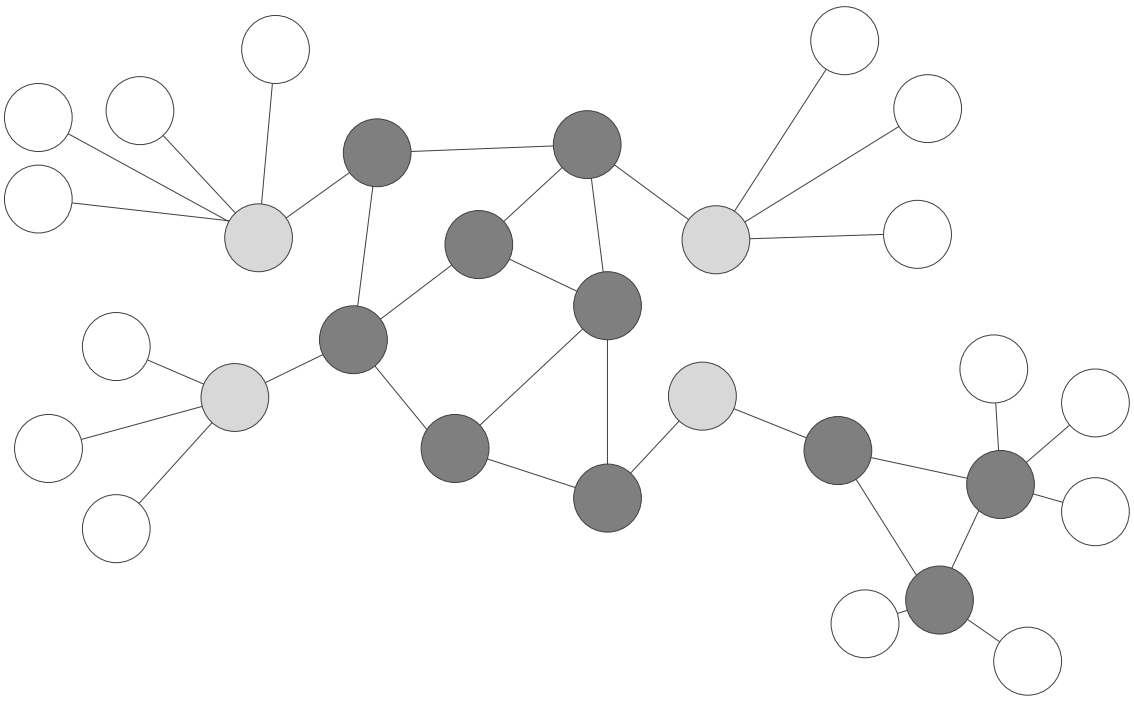
Universidade do Minho

Elementos de Engenharia de Sistemas
Módulo de Optimização de Redes
Problema do Caixeiro Viajante



Filipe Pereira e Alvelos

2016

v0.1

Índice

1 Introdução

1.1 Definição

O problema do caixeiro viajante (PCV, ou *travelling salesman problem* - TSP) consiste em, dada uma rede com distâncias (custos) associadas às arestas, determinar um circuito elementar com menor distância (custo) total que inclua todos os vértices uma e uma só vez.

Dependendo da estrutura de custos, uma instância do PCV pode ser simétrica ou assimétrica. A instância da Figura 1.1 é simétrica porque a distância de um arco é independente do sentido em que este é percorrido (logo a instância pode ser definida numa rede não orientada). Instâncias em que a distância de um arco é diferente da distância do arco que liga os mesmos nodos mas em sentido inverso, dizem-se assimétricas (logo serão necessariamente representadas através de redes orientadas).

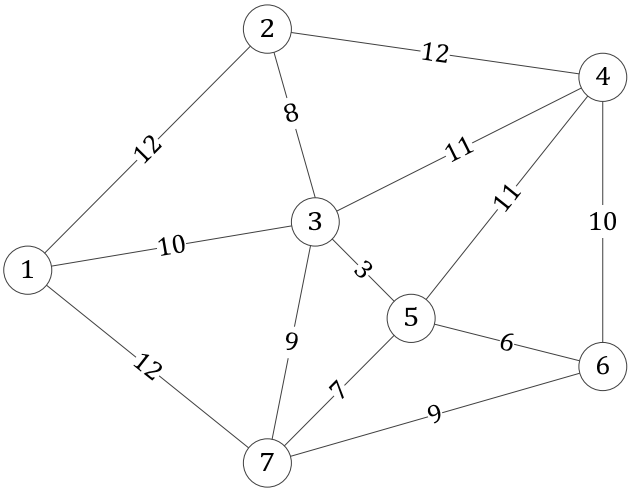


Figura 1.1. Instância do PCV representada por uma rede.

Uma instância do PCV pode também ser caracterizada através de uma matriz de distâncias como a representada na Tabela 1.1. Apenas se considera a matriz triangular superior (excluindo a diagonal) por a instância ser simétrica.

c_{ij}	1	2	3	4	5
1	-	6	3	13	11
2		-	7	10	9
3			-	8	15
4				-	4
5					-

Tabela 1.1. Instância do PCV simétrica representada por uma matriz.

Tipicamente, assume-se que existe um arco entre qualquer par de nodos. Este pressuposto não é restritivo, já que, caso não exista um arco entre dois nodos pode incluir-se um arco artificial com uma distância suficientemente elevada. Por exemplo, na instância da Figura 1.1 seriam incluídos os arcos artificiais 14, 15, 16, 25, 26, 27, 36 e 47, cada um com uma distância suficientemente elevada. A distância de um arco artificial pode ser calculada através do produto do número de vértices pelo maior custo de um arco mais um. No exemplo, 7*12+1=85. Dessa forma, nenhuma solução que inclua arcos artificiais é melhor do que uma solução sem nenhum arco artificial.

Uma solução que inclua arcos artificiais só poderá ser óptima se não existir nenhuma solução admissível sem arcos artificiais (caso contrário seria essa a solução óptima porque a distância de um arco artificial é maior do que a distância de qualquer solução admissível). Assim, se uma solução óptima inclui um arco artificial, o problema original (sem arcos artificiais) é impossível.

Note-se que esta distância artificial deverá ser o mais baixa possível, mas o seu cálculo não deverá ser demasiado elaborado. Uma alternativa ao cálculo agora descrito é usar uma distância artificial duas ordens de grandeza acima da maior distância. No exemplo, a distância artificial seria 1000.

Formalmente, o PCV define-se pelos seguintes parâmetros:

- $G = (N, A)$ grafo orientado (instância simétrica) ou não orientado (instância assimétrica)
- n número de vértices
- c_{ij} distância em que se incorre ao atravessar o arco ij, $\forall ij \in A$
- $c_{ij} \geq 0, \forall ij \in A$

No PCV simétrico: $c_{ij} = c_{ji}$, no PCV assimétrico: $\exists ij \in A: c_{ij} \neq c_{ji}$.

1.2 *PCV relaxado

A definição do PCV inclui a restrição de cada vértice ter de ser visitado uma e uma só vez. Considere-se agora o problema que se obtém ao permitir que um vértice seja visitado mais do que uma vez. Designa-se esse problema por PCV *relaxado*.

No caso de se verificar a desigualdade triangular, $c_{ij} \leq c_{ik} + c_{kj}, \forall i, k, j$, numa solução óptima do PCV relaxado um vértice nunca é visitada mais de uma vez. É esse o caso em que os vértices correspondem a localizações e as distâncias são as distâncias euclideanas.

No caso de não se verificar a desigualdade triangular, pode haver soluções óptimas do PCV relaxado com menor valor do que soluções óptimas PCV. A título exemplificativo, na instância da Tabela 1.2, a solução óptima do PCV (assimétrico) é 1-2-3-1 com custo 22 e a solução óptima do PCV relaxado é 1-2-1-3-1 com custo 4. Note-se que a distância do arco 2-3 é maior do que a soma das distâncias entre 2-1 e 1-3 portanto a desigualdade triangular não se verificar (o caminho mais curto entre 2 e 3 é passar por 1 e não o directo).

	1	2	3
1	-	1	1
2	1	-	20
3	1	-	-

Tabela 1.2. Instância em que o valor óptimo do PCV é maior que o do PCV relaxado.

1.3 Motivação

A relevância do PCV pode ser justificada de diversas formas: i) pela sua importância histórica[1], ii) por ser um problema fundamental (no sentido de desenvolvimentos na sua resolução terem implicações em muitos outros problemas), iii) por ser um problema que tem servido de teste a um elevado número de técnicas de modelação e métodos de optimização, iv) por aparecer como um subproblema em muitos outros problemas e ainda v) pelas aplicações que encontra em problemas reais como, por exemplo, na definição de percursos para a recolha ou entrega de bens em diferentes localizações (e.g. correio, encomendas, notas para reabastecer ATM, armazéns automáticos), no fabrico de circuitos VLSI, no escalonamento de produção e no desenho de redes de computadores[2].

Uma possível abordagem para resolver o PCV é enumerar todas as soluções possíveis, calculando para cada uma o seu valor (distância total) escolhendo a solução com menor valor que, por definição, é a solução óptima. Dado que cada solução corresponde a uma permutação das cidades, o número de soluções possíveis é $n!$ para o PCV assimétrico e $n!/2$ para o PCV simétrico (n é o número de vértices). Esta abordagem é irrealista, mesmo para instâncias pequenas – dezenas de cidades, como se constata pelos valores apresentados na Tabela 1.3.

n (número de cidades)	$n!$ (número de soluções)	Observações
10	3628800	
30	2.65×10^{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.
100	9.3×10^{157}	Estima-se que o número de átomos no universo esteja entre 10^{78} e 10^{82}.

[1] Ver http://www.math.uwaterloo.ca/tsp/history/index.html para uma história do PCV.

[2] Ver http://www.math.uwaterloo.ca/tsp/apps/index.html para detalhes e exemplos adicionais. Em http://www.math.uwaterloo.ca/tsp/pubs/index.html, é apresentada a solução óptima para uma instância em que a cada um de 24727 vértices corresponde um pub do Reino Unido.

Tabela 1.3. Número de soluções para instâncias do PCV com diferentes dimensões.

No entanto, são resolvidas instâncias de muito maior dimensão como as representadas nas Figura 1.2 e Figura 1.3 (ambas com origem em problemas de determinar a sequência pela qual uma máquina deve perfurar um conjunto de pontos numa placa - relevantes no fabrico de circuitos impressos). A instância da Figura 1.3 é a maior instância resolvida até ao momento[3].

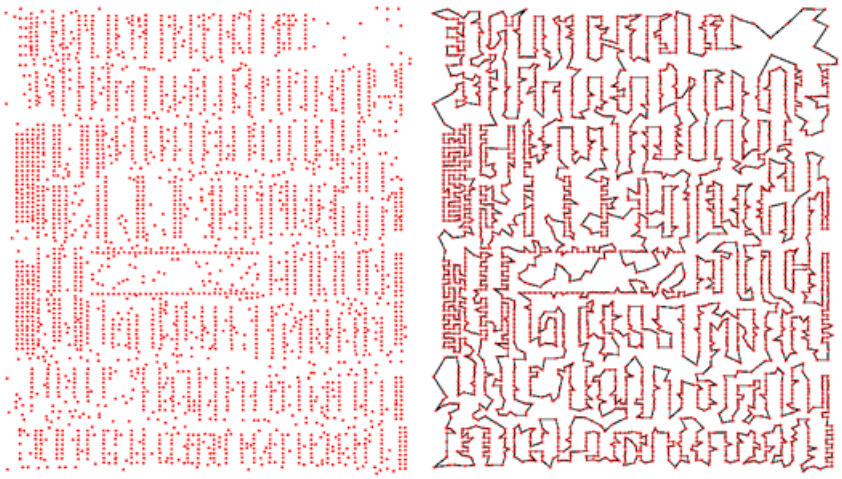


Figura 1.2. Instância e solução óptima de um PCV com 3038 vértices.

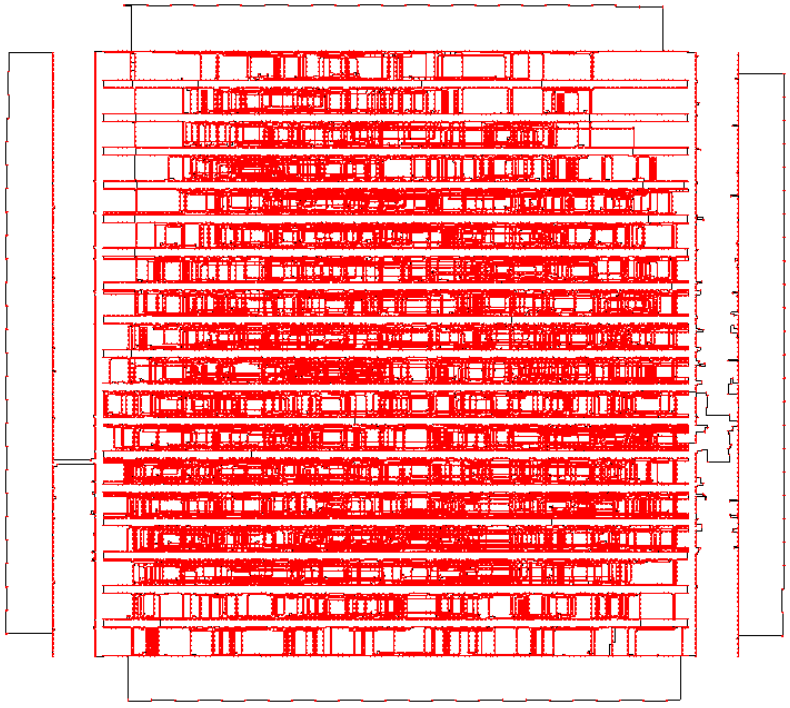


Figura 1.3. Solução óptima de um PCV com 85900 vértices.

[3] http://www.math.uwaterloo.ca/tsp/index.html (de também onde provêm as imagens).

Na base da resolução de problemas desta enorme dimensão estão métodos avançados baseados em programação inteira. Em concreto, na combinação do método de partição (*branch-and-bound*) com cortes e heurísticas.

Existe um prémio de 1000 dólares para quem resolver uma instância do PCV com 100 000 vértices derivada da imagem da "Mona Lisa"[4] (Figura 1.4).

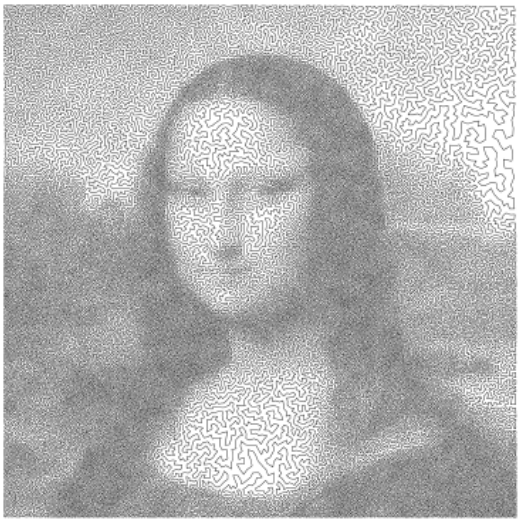


Figura 1.4. Instância "Mona Lisa".

[4] http://www.math.uwaterloo.ca/tsp/data/ml/monalisa.html

2 Heurísticas

Em optimização, uma heurística[5] é um procedimento que se aplica a um problema específico com vista à obtenção (num tempo computacional aceitável) de uma solução de qualidade mas sem nenhuma garantia teórica de tal ser conseguido.

As heurísticas mais relevantes podem ser classificadas em três grupos: heurísticas de construção, heurísticas de pesquisa local e meta-heurísticas. A estes grupos correspondem diferentes níveis de exigências em termos de concepção e implementação (as construtivas são menos exigentes e as meta-heurísticas mais), de rapidez de execução (construtivas mais rápidas, meta-heurísticas menos) e de qualidade das soluções obtidas (construtivas pior, meta-heurísticas melhor).

Para um mesmo problema, podem existir várias heurísticas. O que as diferencia são os princípios gerais que as enquadram (i.e. o seu tipo) e a forma como exploram as características específicas do problema. Dada a elevada liberdade existente na concepção de uma heurística, o papel de quem a concebe é sempre relevante.

As heurísticas estão entre os métodos mais adequados, nalguns casos os únicos, para abordar problemas difíceis, de grandes dimensões, não lineares ou para os quais a obtenção de uma solução num tempo computacional reduzido é crucial.

2.1 Heurísticas construtivas

Uma heurística construtiva parte de uma solução vazia (ou parcial) e adiciona-lhe elementos, um a um, tendo em conta o objectivo do problema e a admissibilidade da solução a obter.

Para além da representação da solução (o que desde logo define os elementos que são considerados), o essencial de uma heurística construtiva reside na forma como é escolhido o elemento que é (tentativamente) adicionado à solução em cada iteração. O mais comum é ser escolhido o elemento que provoca uma menor deterioração no valor da função objectivo de acordo com uma regra simples (para a heurística ser rápida) e razoável para o problema em causa (no sentido de se pretender uma solução de qualidade). Uma característica comum das heurísticas construtivas é serem "gulosas" (*greedy*): optam pelo melhor elemento no passo actual sem olhar às consequências para além deste.

[5] ou método aproximado por oposição a método exacto (método que garante a obtenção de uma solução óptima).

As heurísticas construtivas têm duas qualidades relevantes que facilitam a aceitação da utilização de optimização nalgumas aplicações: podem incluir a tradução de procedimentos humanos para procedimentos computacionais e a sua correcta implementação é facilmente verificável. No entanto, estima-se que as soluções obtidas por heurísticas construtivas fiquem a 10-15% do valor óptimo.

2.1.1 Heurística do vizinho mais próximo[6]

Como o nome indica, a heurística do vizinho mais próximo para o PCV, parte de um vértice e vai formando o circuito seleccionando o vértice mais próximo. Esta heurística baseia-se na representação de uma solução do PCV através de uma permutação. Assim, os elementos adicionados à solução são vértices. O algoritmo é dado de seguida.

```
// Algoritmo do vizinho mais próximo
Considerar um vértice arbitrariamente e o caminho formado apenas por esse
        vértice
Enquanto há vértices que não fazem parte do caminho
    Determinar o vértice mais próximo do último vértice adicionado ao
        caminho (de entre os que não fazem parte do caminho)
    Se não existe, terminar (heurística não obteve solução admissível)
    Se existe, incluí-lo no caminho
Formar o circuito, incluindo no caminho a aresta que liga o último
        vértice considerado ao primeiro
```

Considere-se a instância do PCV da Figura 1.1. Se se aplicar a heurística do vizinho mais próximo partindo o vértice 1, não é obtida nenhuma solução admissível (1-3-5-6-7 e algoritmo pára porque não há nenhum vértice não visitado a partir de 7), o que é consequência da heurística, em cada passo, escolher o melhor elemento sem avaliar as consequências futuras dessa escolha.

Na Figura 2.1 apresenta-se a solução obtida por aplicação da heurística do vizinho mais próximo partindo do vértice 3. A solução (admissível) obtida (3-5-6-7-1-2-4-3) tem um custo total de 65.

[6] Visualização da heurística do vizinho mais próximo em http://www-e.uni-magdeburg.de/mertens/TSP/TSP.html

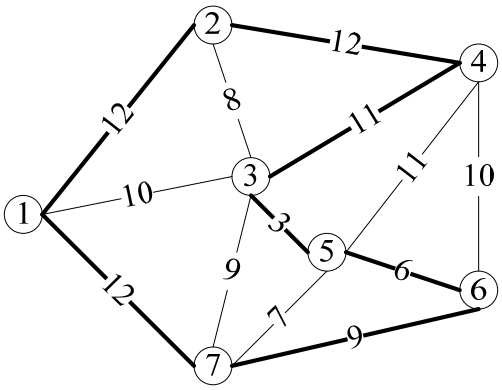


Figura 2.1. Solução dada pela heurística do vizinho mais próximo (partindo de 3).

Considere-se agora a instância da Tabela 1.1. Aplicando a heurística do vizinho mais próximo partindo do vértice 1, como se pode verificar na Tabela 2.1, a solução obtida é 1-3-2-5-4-1 com distância total de 36.

c_{ij}	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**	-

Tabela 2.1. Solução da heurística de menor custo.

Ao contrário do que é comum, existe uma garantia de qualidade para uma solução obtida pela heurística que consiste em aplicar a heurística do vizinho mais próximo n vezes com início em cada um dos n vértices e escolher a melhor solução. Para tal, as distâncias têm de ser todas positivas, a matriz das distâncias ser simétrica e satisfazer a desigualdade triangular, isto é $c_{ij} \leq c_{ik} + c_{kj}, \forall i, \forall j, \forall k$. Para essa heurística e com esses pressupostos, representando z_H como a distância dada pela heurística e por z_{OPT} a distância óptima, verifica-se

$$\frac{z_H}{z_{OPT}} \leq \frac{1}{2}(1 + log_2 n)$$

A extensão da heurística do vizinho mais próximo para o PCV assimétrico é directa.

2.1.2 Heurística da aresta de menor custo

A heurística construtiva da aresta de menor custo para o PCV baseia-se em, em cada iteração, considerar a aresta que tem menor custo (das que ainda não foram consideradas) que não torna a solução não admissível. Assim, uma aresta é candidata a ser adicionada à solução se i) não fizer parte da solução, ii) a sua adição não implicar que um (ou dois) vértice(s) fique(m) com grau 3, iii) a sua adição não criar um subcircuito. Note-se que numa solução do problema do caixeiro viajante, todos os vértices têm grau 2 (cada vértice está ligada a outros dois) e não existem subcircuitos (só há um circuito que é o que inclui todos os vértices uma e uma só vez).

O algoritmo é dado de seguida.

```
// Algoritmo da aresta de menor custo
Enquanto circuito não está completo ou há arestas para considerar
     Seleccionar a aresta com menor custo das ainda não consideradas
          Se adição de aresta não implica a formação de um subcircuito
          nem vértices com grau 3, adicionar aresta à solução
Se arestas seleccionadas não formam circuito, heurística não obteve
     solução
```

Para a instância do PCV da Figura 1.1, a aplicação da aresta de menor custo resulta na consideração dos arcos pela ordem dada na Tabela 2.2, onde também é apresentada a decisão relativa a cada aresta. A aplicação desta heurística nesta instância não resulta numa solução admissível, como se pode constatar pela Figura 2.2.

Iteração	Aresta	Decisão
1	3-5	Incluída na solução
2	5-6	Incluída na solução
3	5-7	Excluída - vértice 5 ficaria com grau 3
4	2-3	Incluída na solução
5	6-7	Incluída na solução
6	3-7	Excluída – formaria subcircuito (3-5-6-7-3) e vértice 3 ficaria com grau 3
7	4-6	Excluída - vértice 6 ficaria com grau 3
8	1-3	Excluída - vértice 3 ficaria com grau 3
9	3-4	Excluída - vértice 3 ficaria com grau 3
10	4-5	Excluída - vértice 5 ficaria com grau 3
11	1-2	Incluída na solução
12	1-7	Excluída - formaria subcircuito
13	2-4	Excluída - vértice 2 ficaria com grau 3

Tabela 2.2. Aplicação da heurística de menor custo.

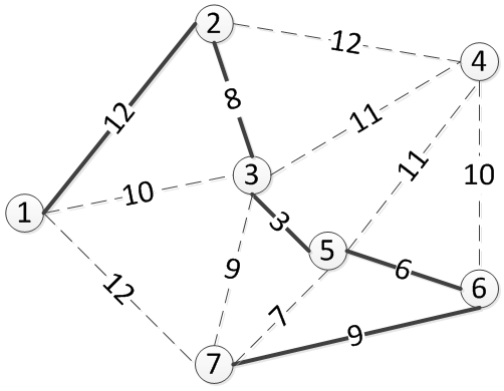


Figura 2.2. Solução dada pela heurística de menor custo.

Considere-se agora a instância da Tabela 1.1. Aplicando a heurística da aresta de menor custo, como se pode verificar nas Tabela 2.3 e Tabela 2.4, a solução obtida é 1-3-4-5-2-1 com distância total de 30.

c_{ij}	1	2	3	4	5
1	-	**6**	**3**	13	11
2		-	*7*	10	**9**
3			-	**8**	15
4				-	**4**
5					-

Tabela 2.3. Instância do PCV simétrico representada por uma matriz.

Iteração	Aresta	Decisão
1	1-3	Incluída na solução
2	4-5	Incluída na solução
3	1-2	Incluída na solução
4	2-3	Excluída - formaria subcircuito (1-2-3)
5	3-4	Incluída na solução
6	2-5	Incluída na solução

Tabela 2.4. Aplicação da heurística da aresta de menor custo.

A extensão da heurística da aresta de menor custo para o PCV assimétrico é quase directa, sendo apenas necessário alterar a exclusão das arestas que implicam vértices com grau 3 para a exclusão de arestas que implicam grau de entrada 2 ou grau de saída 2. Numa solução admissível do PCV simétrico, todos os nodos têm grau de entrada 1 e grau de saída 1.

2.1.3 Heurística de inserção do vértice mais próximo[7]

Na heurística de inserção do vértice mais próximo, a solução inicial é um subcircuito formado pela duplicação da aresta de menor custo. Em cada iteração é inserido um vértice no subcircuito (implicando a remoção de uma aresta e a adição de duas novas arestas) até não ser possível adicionar mais vértices por um circuito (completo) ter sido obtido ou por não existirem arestas que o permitam. O algoritmo é dado de seguida.

[7] Visualização de heurísticas de inserção em http://www-e.uni-magdeburg.de/mertens/TSP/TSP.html

```
// Algoritmo de inserção
Obter o subcircuito i-j-i em que ij é a aresta com menor custo
Enquanto circuito não está completo
     // Seleccionar vértice a inserir
     Seleccionar o vértice k que não faz parte do circuito e que tem a
          menor distância ao subcircuito (a distância do vértice ao
          subcircuito é dada pela menor distância entre o vértice e um
          vértice do circuito)
     // Seleccionar local a inserir
     Seleccionar aresta do circuito ij com menor valor de c_ik + c_kj − c_ij
     Actualizar circuito por remoção da aresta ij e adição das arestas ik
          e kj
```

Considere-se a instância da Tabela 1.1. O circuito inicial é 1-3-1 com distância 6. As distâncias dos vértices que não fazem parte do circuito ao circuito são:

$$d_2 = \min\{6,7\} = 6$$
$$d_4 = \min\{13,8\} = 8$$
$$d_5 = \min\{11,15\} = 11$$

Assim, o vértice escolhido para fazer parte do circuito é o vértice 2 porque tem a menor distância. Uma das arestas 1-3 é removida da solução e são inseridas as arestas 1-2 e 2-3. A variação da distância é $-3 + 6 + 7 = 10$ e a solução passa a valer $6 + 10 = 16$.

O subcircuito passa a ser 1-2-3-1. Actualizando as distâncias:

$$d_4 = \min\{13,8,10\} = 8$$
$$d_5 = \min\{11,15,9\} = 9$$

O vértice escolhido para fazer parte do circuito é o vértice 4 porque tem a menor distância. O vértice 4 pode ser inserido entre os vértices 1 e 2, ou 2 e 3, ou 3 e 1. As variações de custo para cada uma das situações são

$$\Delta_{12}^4 = 13 + 10 - 6 = 17$$
$$\Delta_{23}^4 = 10 + 8 - 7 = 11$$
$$\Delta_{13}^4 = 14 + 8 - 3 = 19$$

Por a variação de custo ser menor, o vértice 4 é inserido entre 2 e 3, passando a solução a ser 1-2-4-3-1 com valor $16 + 11 = 27$.

O vértice 5 pode ser inserido entre 1-2, 2-4, 4-3 e 3-1.

$$\Delta_{12}^5 = 11 + 9 - 6 = 14$$
$$\Delta_{24}^5 = 9 + 4 - 10 = 3$$
$$\Delta_{43}^5 = 15 + 4 - 8 = 11$$
$$\Delta_{13}^5 = 11 + 15 - 3 = 23$$

Por a variação de custo ser menor, o vértice 5 é inserido entre 2 e 4, passando a solução a ser 1-2-5-4-3-1 com valor $27 + 3 = 30$.

Se a instância fosse assimétrica, seria necessário calcular a variação de custo da inserção de cada vértice em cada um dos sentidos.

2.1.4 *Outras heurísticas de inserção

Outras heurísticas de inserção podem ser obtidas alterando a forma como é seleccionado o subcircuito inicial e/ou a forma de selecção do vértice e do local de inserção.

Por exemplo, na heurística de inserção do vértice mais afastado, o vértice seleccionado, como o nome indica, é o que tem a maior distância ao circuito.

Na heurística da inserção mais barata, são calculadas todas as combinações entre arestas do subcircuito e vértices que não fazem parte do subcircuito, seleccionando-se a que tem menor variação.

Na heurística de inserção no invólucro convexo, que apenas faz sentido aplicar em instâncias do PCV euclidianas (e.g. a distância entre dois vértices é a distância euclidiana – que se obtém com base nas coordenadas dos vértices), o subcircuito inicial é obtido com base no invólucro convexo.

O invólucro convexo de um conjunto de pontos X é o conjunto convexo mais pequeno que contém X, tal como ilustrado na Figura 2.3. Na heurística de inserção no invólucro convexo o subcircuito inicial é formado pelos vértices que fazem parte do invólucro convexo. A selecção do vértice k a inserir e da localização da inserção (entre i e j) é feita com base em $\frac{c_{ik}+c_{kj}}{c_{ij}}$.

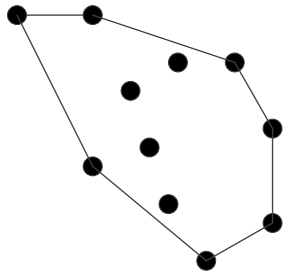


Figura 2.3. Ilustração do conceito de invólucro convexo.

2.2 Heurísticas específicas

Na secção anterior foram apresentadas heurísticas com aspectos específicos do PCV mas que seguem o princípio geral das heurísticas construtivas. Nesta secção apresentam-se duas heurísticas que são totalmente específicas do PCV (embora os mesmos princípios possam estar presentes em heurísticas para problemas relacionados como o problema do caixeiro viajante múltiplo – em que se podem construir vários circuitos - ou problemas de encaminhamento de veículos – o mais simples dos quais se caracteriza por os vértices terem procuras e os veículos capacidades). Refere-se o facto de existirem outras heurísticas específicas, como a Or-opt, Lin-Kernighan, GENI e GENIUS, mais elaboradas e que produzem resultados, em geral, de melhor qualidade. Por exemplo, estima-se que a heurística de Lin-Kernighan obtenha soluções a cerca de 1-2% do valor óptimo.

2.2.1 Heurística de Clarke e Wright

A heurística de Clarke-Wright (também designada por heurística das poupanças) parte de uma solução não admissível que consiste num conjunto de $n-1$ subcircuitos e, em cada iteração, funde dois subcircuitos. Os subcircuitos iniciais são definidos por um vértice central (escolhido arbitrariamente) e cada um dos restantes vértices. A fusão de dois subcircuitos consiste em remover duas arestas, uma de cada subcircuito, que incidem no vértice central e adicionar uma aresta que une os dois vértices (que não o central) que ficam com grau 1. No máximo (quando os dois circuitos têm mais de dois vértices), existem quatro formas de unir dois subcircuitos.

Em cada iteração é calculada a variação de custo (inverso da poupança) das possíveis fusões de cada par de subcircuitos e é efectuada a fusão com a menor variação de custo (maior poupança).

A poupança de unir dois subcircuitos em que i faz parte de um subcircuito e j faz parte do outro subcircuito e ambos estão ligados ao nodo central é

$$s_{ij} = \begin{cases} c_{ic} + c_{cj} - c_{ij}, \; se\ a\ aresta\ ij\ \text{é}\ admissível \\ -\infty, se\ a\ aresta\ ij\ não\ \text{é}\ admissível \end{cases}$$

```
// Algoritmo de Clarke e Wright
Seleccionar arbitrariamente um nodo central c e adicionar à solução as
        arestas entre c e cada um dos restantes nodos duas vezes
Enquanto não existir apenas um circuito
    Para cada par de subcircuitos c-i1-...-j1-c e c-i2-...-j2-c,
        considerar a poupança para
        remoção das arestas c-i1 e c-i2 e adição da aresta i1-i2
        remoção das arestas c-i1 e c-j2 e adição da aresta i1-j2
        remoção das arestas c-j1 e c-i2 e adição da aresta j1-i2
        remoção das arestas c-j1 e c-j2 e adição da aresta j1-j2
    Fundir o par de subcircuitos com maior poupança
```

O cálculo inicial da matriz das poupanças torna o algoritmo mais eficiente.

Considere-se um exemplo da Tabela 2.5 com cinco vértices. O vértice 1 é escolhido arbitrariamente para vértice central.

Os subcircuitos iniciais são 1-2-1 (custo 12), 1-3-1 (custo 6), 1-4-1 (custo 26) e 1-5-1 (custo 22) com custo total de 66.

c_{ij}	1	2	3	4	5
1	-	6	3	13	11
2		-	7	10	9
3			-	8	15
4				-	4
5					-

Tabela 2.5. Instância do PCV simétrica representada por uma matriz.

As poupanças são dadas na Tabela 2.6. Por exemplo, a poupança na célula 2-3 corresponde a fundir os subcircuitos 1-2-1 e 1-3-1. Tal implica a remoção das arestas 1-2 e 1-3 e a

adição da aresta 2-3. A poupança é $6 + 3 - 7 = 2$. Note-se que se os subcircuitos 1-3-1 e 1-5-1 forem fundidos, a solução obtida tem um custo maior do que a actual.

s_{ij}	1	2	3	4	5
1	-	-	-	-	-
2	-	-	2	9	8
3	-		-	8	-1
4	-			-	20
5	-				-

Tabela 2.6. Poupanças do exemplo.

A fusão escolhida é a dos subcircuitos 1-4-1 e 1-5-1, passando a fazer parte da solução o subcircuito 1-4-5-1. O custo da nova solução é $66 - 20 = 46$.

As possíveis fusões são agora 1-2-1 com 1-3-1, 1-2-1 com 1-4-5-1 e 1-3-1 com 1-4-5-1. O maior valor da tabela das poupanças (excluindo células de pares de vértices no mesmo subcircuito. i.e. a célula 4-5) é 9 e corresponde a fundir os circuitos 1-2-1 e 1-4-5-1 removendo as arestas 1-2 e 1-4 e adicionando a aresta 2-4. O subcircuito 1-2-4-5-1. O custo da nova solução é $46 - 9 = 37$. Resta considerar a fusão do circuito 1-3-1 com o circuito 1-2-4-5-1 que pode ser feita de duas formas, adicionando a aresta 2-3 ou adicionando a aresta 3-5. Sendo a poupança maior no primeiro caso, o circuito obtido é 1-3-2-4-5-1 com custo $37 - 2 = 35$.

Para instâncias assimétricas, seria necessário ter em conta a orientação dos subcircuitos.

2.2.2 Heurística da árvore de suporte de custo mínimo

O problema da árvore de suporte de custo mínimo define-se numa rede não orientada em que existe um custo associado a cada aresta. Uma árvore de suporte é um conjunto de $n - 1$ arestas (em que n é o número de vértices da rede) que não forma (sub)circuitos. É possível determinar uma solução óptima para este problema através da aplicação, entre outros, do algoritmo de Kruskal que consiste em ordenar as arestas por ordem crescente de custo e considerando um arco de cada vez, testar se a sua inclusão na solução forma um (sub)circuito. Se a inclusão do arco não formar um (sub)circuito, o arco é incluído na árvore de suporte; caso contrário, passa-se para a aresta seguinte. Note-se que não é frequente um algoritmo construtivo, como o agora descrito, garantir a obtenção de uma solução óptima.

Dada a existência de algoritmos muito eficientes e que obtêm a solução óptima para o problema da árvore de suporte de custo mínimo (como o de Kruskal agora referido) e a semelhança com o problema do caixeiro viajante – em ambos os problemas pretende-se minimizar o custo de um conjunto de arestas, embora este conjunto defina objectos diferentes), parece razoável a estratégia de primeiro obter uma árvore de suporte de custo mínimo e em seguida transformá-la numa solução para o problema do caixeiro viajante.

Assim, num primeiro passo, obtém-se a árvore de suporte de custo mínimo. Num segundo passo, cada aresta da árvore de suporte de custo mínimo é duplicada. Em seguida, gera-se um circuito Euleriano[8] que existe sempre por o número de arestas ser par de acordo com o teorema de Euler. Este circuito é transformado numa solução do PCV (um circuito Hamiltoniano): percorrendo-se o circuito Euleriano, os vértices que ainda não foram visitados são adicionados ao circuito Hamiltoniano.

```
// Algoritmo da heurística da árvore de suporte de custo mínimo
Obter árvore de suporte de custo mínimo
Duplicar todas as arestas da árvore de suporte de custo mínimo
Obter um circuito Euleriano
Transformar o circuito Euleriano num circuito Hamiltoniano percorrendo o
         circuito Euleriano sem repetição de arcos e adicionando ao
         circuito Hamiltoniano, pela mesma ordem, os vértices que ainda
         dele não fazem parte
```

A título exemplificativo, considere-se a instância da Tabela 2.7.

c_{ij}	1	2	3	4	5
1	-	**6**	**3**	**8**	11
2		-	7	10	9
3			-	9	11
4				-	**4**
5					-

Tabela 2.7. Instância do PCV simétrico.

A aplicação do algoritmo de Kruskal é dada na Tabela 2.8 e resulta na árvore de suporte de custo mínimo 1-2, 1-3, 1-4, 4-5.

Iteração	Aresta	Decisão
1	1-3	Incluída na solução
2	4-5	Incluída na solução
3	1-2	Incluída na solução
4	2-3	Excluída - formaria subcircuito (1-2-3)
5	1-4	Incluída na solução

Tabela 2.8. Aplicação do algoritmo de Kruskal.

O passo seguinte é duplicação das arestas. O circuito Euleriano, tal como representado na Figura 2.4, é 1-4-5-4-1-3-1-2-1. O circuito Hamiltoniano, representado pelos arcos a negro na Figura 2.5 é então 1-4-5-3-2-1.

[8] Circuito Euleriano - de Euler (1707-1783) - é um circuito que inclui todos os arcos uma e uma só vez (os 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.

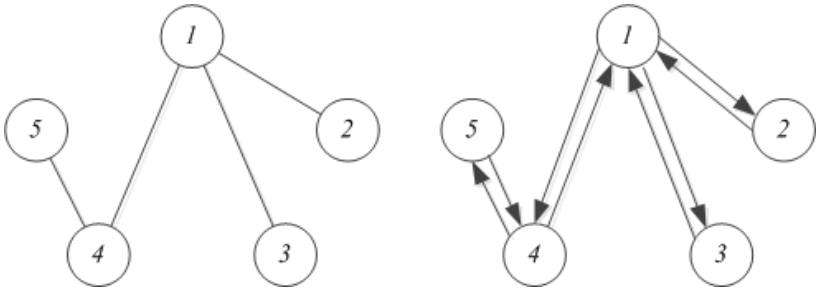


Figura 2.4. Árvore de suporte de custo mínimo e caminho euleriano.

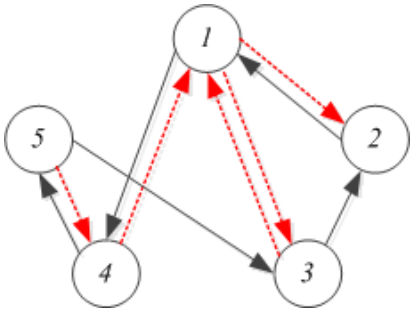


Figura 2.5. Construção da solução do PCV com a heurística da árvore de suporte de custo mínimo.

Um melhoramento desta heurística (a heurística de Christofides) obtém o circuito Euleriano através da resolução de um problema de emparelhamento perfeito de custo mínimo associado aos nodos com grau ímpar da árvore (e não por substituição das arestas por dois arcos de sentidos opostos).

2.3 Heurísticas de pesquisa local

O conceito fundamental de uma heurística de pesquisa local é o conceito de vizinhança. A vizinhança de uma solução é o conjunto de soluções que se podem obter a partir dela através de uma modificação elementar. Essencialmente, o que diferencia uma heurística de pesquisa local de outra para o mesmo problema é a estrutura de vizinhança.

Um algoritmo genérico de uma heurística de pesquisa local (para um problema de minimização) é o seguinte.

```
// Algoritmo de pesquisa local genérico
Obter uma solução inicial x (através de uma heurística construtiva,
         aleatoriamente, ou de outra forma)
// Melhorou é uma variável booleana
Melhorou=true
Enquanto Melhorou==true
     Avaliar todas as soluções vizinhas de x e guardar a melhor y
     Se f(y) < f(x), x = y
     Se não, Melhorou=false
```

O algoritmo apresentado é designado por máxima descida por que é escolhida a melhor solução vizinha. Alternativamente, pode ser escolhida a primeira solução que é melhor do que a actual. Uma variante estocástica do algoritmo de pesquisa local consiste em fazer uma amostra da vizinhança de forma aleatória (e, tal como na pesquisa determinística, escolher a melhor vizinha ou a primeira melhor do que a actual, de entre as consideradas).

No final da aplicação do algoritmo de pesquisa local (determinístico) a solução obtida é um óptimo local (por oposição a global) para a estrutura de vizinhança utilizada.

2.3.1 Troca de duas arestas (2-Opt)[9]

A estrutura de vizinhança de troca de duas arestas para o PCV, também conhecida por 2-Opt, consiste em remover duas arestas não adjacentes do circuito e reconstruir o circuito com outras duas arestas. Na Figura 2.6 mostram-se as cinco soluções vizinhas (de acordo com a estrutura de vizinhança de troca de duas arestas) de uma solução de um PCV com cinco vértices. Note-se que, após removidas duas arestas só há duas arestas que, ao serem inseridas, fazem com que se forme um circuito.

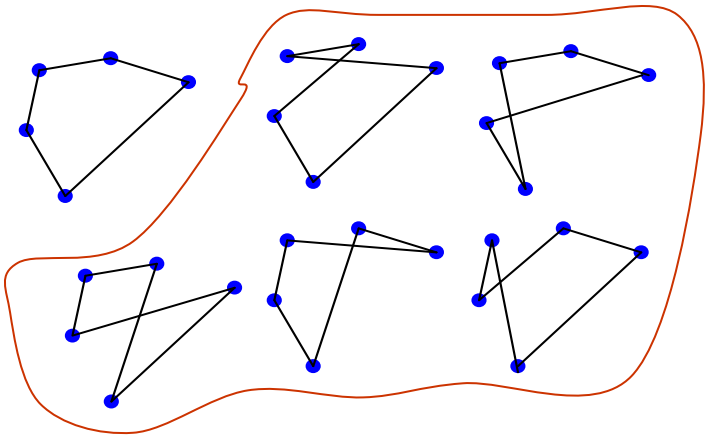


Figura 2.6. As cinco soluções vizinhas de uma solução num PCV com cinco vértices.

[9] Visualização da heurística 2-Opt em http://www-e.uni-magdeburg.de/mertens/TSP/TSP.html

Se se representar de uma solução do PCV como uma permutação (sequência de números de 1 a n sem repetições), obtém-se uma solução vizinha 2-Opt pode ao inverter-se uma subsequência (de comprimento até $n-2$). Esta forma expedita de obter soluções vizinhas 2-Opt é agora exemplificada com a instância da Tabela 2.9.

c_{ij}	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	-

Tabela 2.9. Instância do PCV simétrico.

Aplicando a heurística do vizinho mais próximo partindo do vértice 1 obtém-se a solução 1-3-2-5-4-1 com distância total de 36. A primeira iteração da heurística da troca de duas arestas é dada nas Tabela 2.10, assumindo uma estratégia de pesquisa de máxima descida (determinística) a solução actual da iteração seguinte é a solução 1-3-4-5-2 (solução vizinha 4) por ter o menor custo e ser melhor do que a solução actual.

						Custo	Arestas removidas	Arestas inseridas	Subsequência invertida
Solução actual	1	3	2	5	4	36	-	-	-
Solução vizinha 1	1	**2**	**3**	5	4	45	1-3 e 2-5	1-2 e 3-5	3-2
Solução vizinha 2	1	**5**	**2**	**3**	4	48	1-3 e 5-4	1-5 e 3-4	3-2-5
Solução vizinha 3	1	3	**5**	**2**	4	50	3-2 e 5-4	3-5 e 2-4	2-5
Solução vizinha 4	1	3	**4**	**5**	**2**	30	3-2 e 4-1	3-4 e 1-2	2-5-4
Solução vizinha 5	1	3	2	**4**	**5**	35	2-5 e 4-1	2-4 e 1-5	5-4

Tabela 2.10. Primeira iteração da heurística da troca de duas arestas.

Na Tabela 2.11 é dada a informação relativa à segunda iteração da heurística da troca de duas arestas. Como todas as soluções que se obtêm por troca de duas arestas são piores (ou iguais) do que solução a actual, a solução actual é uma solução óptima local (relativamente à estrutura de vizinhança de troca de duas arestas) e o algoritmo termina.

						Custo
Solução actual	1	3	4	5	2	30
Solução vizinha 1	1	**4**	**3**	5	2	51
Solução vizinha 2	1	**5**	**4**	**3**	2	36
Solução vizinha 3	1	3	**5**	**4**	2	38
Solução vizinha 4	1	3	**2**	**5**	**4**	36
Solução vizinha 5	1	3	4	**2**	**5**	41

Tabela 2.11. Segunda iteração da heurística da troca de duas arestas.

2.3.2 Troca de três arestas (3-Opt)

Na estrutura de vizinhança de 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 (em que pelo menos uma é diferente das removidas). Na Figura 2.7 estão representadas as sete formas de inserir arestas depois de definidas as três arestas que são removidas de uma solução. Quatro vizinhas têm três arestas diferentes da solução actual e três soluções têm apenas duas arestas diferentes da solução actual.

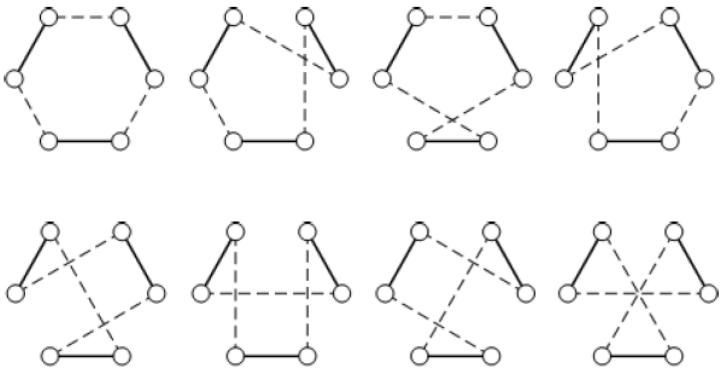


Figura 2.7. Solução actual (canto superior esquerdo) e sete soluções vizinhas 3-Opt obtidas pela remoção das três arestas a tracejado na solução actual.

Estima-se que a heurística 3-Opt obtenha soluções a 3-4% do valor óptimo.

2.3.3 Troca de dois vértices

Na estrutura de vizinhança de troca de dois vértices, como o nome indica, a troca de posição de dois vértices de uma solução. Por exemplo, a solução 1-2-3-4-5 terá como vizinhas as soluções 2-1-3-4-5, 3-1-2-4-5, 4-1-2-3-5, ... Numa instância simétrica com n vértices o número de vizinhas é $n(n-1)/2$.

Na Figura 2.8 é apresentado um exemplo de uma solução actual 1-2-3-4-5-6-7 (custo 69) e da solução vizinha que se obtém por troca das cidades 3 e 4, ou seja 1-2-4-3-5-6-7 (custo 65).

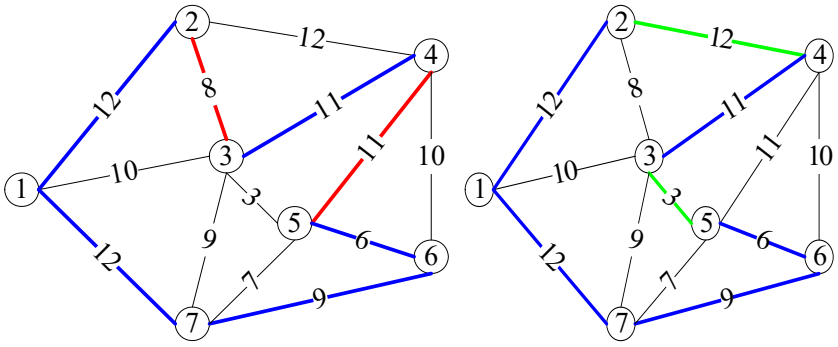


Figura 2.8. Exemplo de solução vizinha por troca de dois vértices.

2.4 *Meta-heurísticas[10]

A ideia fundamental das meta-heurísticas é ter um procedimento a um nível acima (meta) da heurística (em geral de pesquisa local) que permita que o encontrar de soluções de qualidade (em particular óptimos locais) uma solução óptima local não termine com a exploração do espaço de soluções. Cada meta-heurística tem os princípios que a caracterizam mas também componentes específicas do problema, usualmente incorporados na pesquisa local.

2.4.1 GRASP (Greedy Adaptive Search Procedure)

Numa meta-heurística GRASP, em cada iteração é aplicada uma heurística construtiva (aleatorizada) seguida de pesquisa local. Na sua versão mais básica, as iterações são independentes entre si, podendo o GRASP ser visto como uma heurística de pesquisa local com múltiplos inícios (multi-start local search - pesquisa local aplicada várias vezes mas com soluções iniciais diferentes).

Em cada iteração da heurística construtiva não é necessariamente seleccionado o melhor elemento para entrar na solução (como nas heurísticas construtivas "greedy") mas é escolhido um elemento aleatoriamente de entre os que pertencem a uma lista restrita de candidatos (RCL – restricted candidate list) que contém os melhores elementos.

---

[10] Na continuidade das heurísticas apresentadas, descrevem-se duas meta-heurísticas cuja principal característica é a relação entre componentes construtivas e de vizinhanças (GRASP e VND). A pesquisa tabu introduz o conceito de memória. Outros aspectos importantes em meta-heurísticas como a aleatoriedade (usado no GRASP e no VNS, e que tem um papel na meta-heurística clássica pesquisa por arrefecimento simulado aqui não abordada) e a utilização de populações (como nos algoritmos genéticos ou no pesquisa por dispersão) são deixados para um aprofundamento do tópico.

Um elemento j faz parte da RCL se o seu custo não estiver acima de uma determinada proporção da amplitude entre o maior e menor custo (em minimização): $c_j \leq c_{min} + \alpha(c_{max} - c_{min})$. Desta forma garante-se que os elementos com o melhor valor nunca são excluídos.

O parâmetro α, $\alpha \in [0,1]$, regula o tamanho da RCL e o grau de aleatoriedade da selecção. Se $\alpha = 0$, o GRASP é equivalente a uma heurística construtiva seguida de pesquisa local. Se $\alpha = 1$, o GRASP é próximo de obter uma solução aleatória para inicializar uma pesquisa local.

O algoritmo é dado de seguida.

```
Enquanto critério de paragem não for atingido
     // Algoritmo construtivo
     Começar com solução vazia
     Construir conjunto de elementos candidatos a entrar na solução e
          avaliá-los
     Repetir até não haver elementos candidatos
          Definir uma Lista Restrita de Candidatos (RCL)
          Seleccionar aleatoriamente um elemento da RCL
          Adicionar o elemento seleccionado à solução
          Definir o conjunto de elementos candidatos a entrar na solução
               e avaliá-los
     // Pesquisa local
     Partindo da solução obtida, aplicar pesquisa local
```

Qualquer das heurísticas construtivas e das heurísticas de pesquisa local apresentadas para o PCV podem ser usadas como componentes de uma meta-heurística GRASP.

2.4.2 Vizinhanças variáveis

Nesta subsecção descrevem-se duas meta-heurísticas próximas da pesquisa local: descida em vizinhanças variáveis (VND - variable neighborhood descent) e pesquisa em vizinhanças variáveis (VNS - variable neighborhood search).

O conceito fundamental destas abordagens é usar mais do que uma vizinhança. As vizinhanças são ordenadas das mais pequenas (menos modificações na solução actual e pesquisadas de forma mais eficiente) para as maiores.

No VND, começa-se por efectuar uma pesquisa local de acordo com uma estrutura de vizinhança. Quando é obtido um óptimo local para a primeira estrutura de vizinhança, passa-se a usar uma segunda estrutura de vizinhança. Se há uma actualização da solução actual, regressa-se à estrutura de vizinhança actual.

```
Obter solução inicial
Vizinhança actual é a primeira
Enquanto a vizinhança actual não for a última
     Escolher melhor vizinha de acordo com a vizinhança actual
     Se vizinha é melhor do que actual, vizinhança actual passa a ser a
          primeira
     Se não, vizinhança actual passa a ser a vizinhança seguinte
```

No VNS apenas a primeira estrutura de vizinhança é usada para efectuar pesquisa local. As restantes são usadas para seleccionar (de forma aleatória) uma solução a partir da qual se aplica a pesquisa local.

No caso do PCV, as estruturas de vizinhança 2-Opt e 3-Opt, dada a sua estrutura hierárquica, são bons pontos de partida para a aplicação de VND e VNS.

2.4.3 Pesquisa tabu

Na pesquisa tabu, ao contrário do que acontece na pesquisa local, uma solução vizinha pode ser seleccionada para solução actual mesmo sendo pior do que esta. Na sua versão mais simples, a pesquisa tabu inclui um mecanismo de memória (a lista tabu) que evita a entrada em ciclo, permitindo que a pesquisa seja diversificada.

Na lista tabu são guardados os movimentos mais recentes, em que um movimento é a alteração efectuada a uma solução. Por exemplo, na vizinhança 2-Opt um movimento corresponde a duas arestas que são removidas.

Assim, em cada iteração da pesquisa tabu, é seleccionada a melhor solução vizinha de entre as que (i) não são tabu ou (ii) são tabu mas respeitam critério de aspiração (por exemplo, serem melhores do que a solução incumbente, i.e., a melhor solução encontrada até ao momento).

O algoritmo é dado de seguida.

```
Obter uma solução inicial
Repetir
     Gerar vizinhança da solução actual
     Escolher melhor solução da vizinhança tal que: solução não é tabu ou
          é tabu mas satisfaz o critério de aspiração
     Actualizar lista tabu por remoção do movimento mais antigo e
          inserção do movimento mais recente
     Se a solução actual é melhor do que a solução incumbente, actualizar
          solução incumbente
Até ser atingido um número de iterações sem melhoria da incumbente (ou um
          número de iterações limite ou um tempo limite)
```

3 *Programação inteira

Excepto quando explicitamente indicado, nesta secção, consideram-se instâncias do PVC assimétrico.

Todos os modelos usam as variáveis de decisão x_{ij} que definem um circuito

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

e todos os modelos usam as seguintes restrições que forçam a que todos os nodos tenham grau de entrada 1 (um arco a entrar) e um grau de saída 1 (um arco a sair):

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

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

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

A função objectivo é também a mesma em todos os modelos:

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

3.1 Dantzig-Fulkerson-Johnson

No modelo de Dantzig-Fulkerson-Johnson consideram-se adicionalmente as restrições que eliminam subcircuitos:

$$\sum_{ij \in A(S)} x_{ij} \leq |S| - 1, \forall S \subset N\,, |S| \geq 2$$

Para um conjunto de nodos S, $A(S)$ é o conjunto dos arcos que têm ambas as extremidade em nodos de S. O número de nodos de S (o cardinal do conjunto S) é representado por $|S|$.

Existe uma restrição de eliminação de subcircuitos para cada subconjunto de nodos S. A restrição para um subconjunto de nodos S indica que o número de arcos entre nodos de S tem de ser menor do que o número de nodos de S. Assim, um conjunto de arcos que forme um circuito envolvendo todos os nodos de S não é admissível. Por exemplo, numa instância com seis nodos numa rede completa, $N = \{1,2,3,4,5,6\}$, ao conjunto de nodos $S = \{1,2,3\}$, corresponde $A(S) = \{12,13,21,23,31,32\}$ e a restrição que elimina o subcircuito 1-2-3 e o subciruito 3-2-1 é $x_{12} + x_{13} + x_{21} + x_{23} + x_{31} + x_{32} \leq 2$.

Dado que o número de subconjuntos de nodos cresce exponencialmente com o aumento da rede, a resolução directa do modelo de Dantzig-Fulkerson-Johnson não é praticável em instâncias de alguma dimensão. Usualmente é aplicado o algoritmo seguinte:

```
Resolver o problema sem as restrições de eliminação de subcircuitos
Enquanto a solução contiver subcircuitos
     Identificar os nodos dos subcircuitos
     Para cada subcircuito, definir S como o conjunto dos nodos do
          subcircuito e adicionar a restrição correspondente ao problema
     Resolver problema obtido
```

A título exemplificativo, considere-se a instância do problema do caixeiro viajante definida pela matriz de custos.

c_{ij}	*1*	*2*	*3*	*4*	*5*	*6*
1	100	4	5	4	8	6
2	5	100	3	10	6	9
3	5	10	100	1	3	10
4	3	5	1	100	8	6
5	10	10	10	8	100	7
6	1	8	2	3	5	100

O modelo sem restrições de eliminação de subcircuitos é

$$Min\ z = 4x_{11} + 5x_{12} + \cdots + 5x_{65}$$
$$sujeito\ a:$$
$$x_{12} + x_{13} + \cdots + x_{16} = 1$$
$$x_{21} + x_{23} + \cdots + x_{26} = 1$$
$$\ldots$$
$$x_{61} + x_{62} + \cdots + x_{65} = 1$$
$$x_{21} + x_{31} + \cdots + x_{61} = 1$$
$$\ldots$$
$$x_{61} + x_{26} + \cdots + x_{56} = 1$$
$$x_{ij} \in \{0,1\}, \forall ij \in A$$

Uma solução óptima deste modelo (sem as restrições de eliminação de subcircuitos) é $x_{12} = x_{25} = x_{56} = x_{61} = x_{34} = x_{43} = 1$ e todas as restantes variáveis com valor zero. O valor da solução é 20.

Esta solução tem dois subcircuitos: $1 - 2 - 5 - 6 - 1$ e $3 - 4 - 3$. Para eliminar cada um destes dois subcircuitos. adicionam-se as restrições (para $S = \{1,2,5,6\}$ e $S = \{3,4\}$):

$$x_{12} + x_{15} + x_{16} + x_{21} + x_{25} + x_{26} + x_{51} + x_{52} + x_{56} + x_{61} + x_{62} + x_{65} \leq 3$$
$$x_{34} + x_{43} \leq 1$$

A nova solução óptima tem valor 23 e é $x_{12} = x_{23} = x_{35} = x_{56} = x_{64} = x_{41} = 1$. Dado não ter subcircuitos, esta solução é óptima do problema original (em que se consideram todas as restrições de eliminação de subcircuitos).

3.2 Miller-Tucker-Zemlin

O modelo de Miller-Tucker-Zemlin difere na forma como são eliminados os subcircuitos. Neste modelo consideram-se variáveis adicionais associadas à sequência pela qual as cidades são visitadas. A primeira cidade da sequência é a cidade 1. As variáveis contínuas (u) definem a sequência

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

Tendo em conta que $x_{ij} = 1$ implica que $u_j \geq u_i + 1$ (j é visitado depois de i), o grupo de restrições

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

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

garante a eliminação de subcircuitos.

Note-se que $x_{ij} = 1$ implica $u_j - u_i \geq 1$, i.e., o nodo j é forçado a estar depois do nodo i e que $x_{ij} = 0$ implica $u_j - u_i \geq -n + 1$, i.e., a restrição é redundante.

Para o exemplo introduzido anteriormente, uma solução óptima com valor 23 é $x_{12} = x_{25} = x_{56} = x_{63} = x_{34} = x_{41} = 1$ e $u_2 = 0,\ u_3 = 3,\ u_4 = 4,\ u_5 = 1, u_6 = 2$.

A grande vantagem deste modelo em relação ao anterior é o número de restrições de eliminação de subcircuitos ser muito menor, sendo próximo do número de arcos. Tal permite a resolução directa (não iterativa) do problema.

3.3 Fluxos

Apresentam-se dois modelos de fluxos, um modelo agregado e um modelo desagregado.

O modelo de fluxos (agregados) tem como e as variáveis de decisão adicionais

$$y_{ij} - fluxo\ no\ arco\ ij, \forall ij \in A$$

e as restrições adicionais

$$\sum_{1j \in A} y_{1j} = n - 1$$

$$\sum_{j:ij \in A} y_{ij} - \sum_{j:ji \in A} y_{ji} = -1\ , \forall i \in N \setminus \{1\}$$

$$y_{ij} \leq (n-1)x_{ij}, \forall ij \in A$$

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

$$y_{ij} \geq 0, ij \in A$$

O primeiro e segundo grupos de restrições forçam a que saia uma unidade de fluxo do nodo 1 para cada um dos nodos restantes. O terceiro grupo de restrições garante que se existe fluxo num arco, este arco tem de fazer parte do circuito (que é definido pelas variáveis x_{ij}).

No modelo de fluxo desagregado, as variáveis y_{ij} são desagregadas por destino do fluxo:

$$y_{ijk} - fluxo\ no\ arco\ ij\ com\ destino\ a\ k, \forall ij \in A,\ \forall k \in N \setminus \{1\}$$

As restrições adicionais são

$$\sum_{j:ij\in A} y_{ijk} - \sum_{j:ji\in A} y_{jik} = \begin{cases} 1 \; se \; i = 1 \\ 0 \; se \; i \neq k \; e \; i \neq 1, \forall i \in N, \forall k \in N \setminus \{1\} \\ -1 \; se \; i = k \end{cases}$$

$$y_{ijk} \leq x_{ij}, \forall ij \in A, \forall k \in N \setminus \{1\}$$

$$y_{ij} \geq 0, ij \in A$$

O primeiro e segundo grupos de restrições forçam a que saia uma unidade de fluxo do nodo 1 para cada um dos nodos restantes. O terceiro grupo de restrições garante que se existe fluxo num arco, este arco tem de fazer parte do circuito (que é definido pelas variáveis x_{ij}).

3.4 Etapas

No modelo de etapas, as variáveis de decisão adicionais são

$$y_{ijt} = \begin{cases} 1, se \; arco \; ij \; é \; atravessado \; na \; etapa \; t \\ 0, \; caso \; contrário \end{cases}, \forall ij \in A, t = 1, \dots, n$$

As restrições adicionais são

$$\sum_{t\geq 2} \sum_{j:ij\in A} t y_{ijt} - \sum_{t} \sum_{j:ji\in A} t y_{jit} = 1, \; \forall i \in N \setminus \{1\}$$

$$\sum_{j:1j\in A} y_{1j1} = 1 \, , \forall ij \in A$$

$$\sum_{j:j1\in A} y_{j1n} = 1 \, , \forall ij \in A$$

$$x_{ij} = \sum_{t} y_{ijt} \, , \forall ij \in A$$

$$y_{ijt} \geq 0, \forall ij \in A, t = 1, \dots, n$$

O primeiro grupo de restrições indica que se um arco entra no nodo i (excluindo o nodo 1) na etapa t, então o arco da etapa seguinte sai de i (etapa $t + 1$). O segundo e terceiro grupos de restrições asseguram que uma arco sai de 1 na primeira etapa e chega a 1 na última etapa. O quarto grupo de restrições assegura que se um arco é usado então tem de ser usado numa etapa e vice-versa.

3.5 Comparação entre modelos

Diferentes modelos têm comportamentos computacionais muito diferentes, o que deriva qualidade dos limites inferiores dados pela relaxação linear do modelo. O modelo de fluxos desagregado tem melhor comportamento computacional do que o modelo de fluxos agregado. O modelo de etapas tem um muito mau comportamento computacional.

4 *Problemas do caixeiro viajante com lucros

4.1 Definição

Nesta secção referem-se problemas relacionados com o problema do caixeiro viajante mas em que não é necessário visitar todos os vértices (pretende-se portanto um subcircuito) mas existe um lucro associado a visitar cada vértice.

A notação é a seguinte

- $G = (N, A)$ grafo orientado (instância simétrica) ou não orientado (instância assimétrica)
- n número de vértices
- c_{ij} distância em que se incorre ao atravessar o arco ij, $\forall ij \in A$
- $c_{ij} \geq 0, \forall ij \in A$
- p_i lucro associado a visitar o vértice $i, \forall i \in N$

Assume-se que o vértice 1 tem de ser visitado.

Existem dois critérios para avaliar a qualidade de uma solução: a distância percorrida e o lucro obtido. De forma natural, surge o problema bi-objectivo com os objectivos da minimização da distância do (sub)circuito e da maximização do lucro dos nodo do (sub)circuito.

No entanto, frequentemente, este problema bi-objectivo é tratado através de uma das três simplificações seguintes (que o transformam num problema de objectivo único):

- objectivo único é a soma ponderada dos dois objectivos (*profitable tour problem - PTP*);
- pretende-se maximizar o lucro com uma restrição de distância máxima (c_{max}) (*orienteering problem - OP*);
- pretende-se minimizar a distância com o lucro com uma restrição de lucro mínimo (p_{min}) (*prize collecting problem - PCP*).

4.2 Abordagens heurísticas

A heurística construtiva mais natural para problemas do caixeiro viajante com lucros é baseada na inserção de um vértice. Partindo do vértice 1, é inserido o vértice que mais aumenta a soma ponderada dos objectivos (PTP), mais aumenta o lucro (OP) ou o vizinho mais próximo (PCP). A heurística pára quando não é possível aumentar a soma ponderada dos dois objectivos (PTP), a inserção de qualquer vértice excede a distância máxima (OP) ou o lucro mínimo foi atingido (PCP). É de referir que deverá ser tido em conta que a inserção de um vértice pode ser

feita em diferentes posições do (sub)circuito. Se o vértice k for inserido entre i e j, a variação da distância é $c_{ik} + c_{kj} - c_{ij}$ e a variação do lucro é p_k.

Uma heurística simétrica à agora descrita (poder-se-á chamar heurística destrutiva) consiste em obter uma solução do PCV (possivelmente heurística) e remover vértices iterativamente tendo em conta a variação da distância e do lucro de forma similar à heurística construtiva.

Dado que uma solução de um problema do problema do caixeiro viajante com lucros não tem um número de vértices fixo (ao contrário de uma solução para o problema do caixeiro viajante), as duas operações descritas (inserção e remoção) podem ser usadas para definir (sub)vizinhanças. Outra (sub)vizinhança consiste na substituição de um vértice do (sub)circuito por um vértice que não está no (sub)circuito (mantendo o número de vértices do (sub)circuito).

Com a união destas três operações (inserção, remoção e substituição) garante-se uma importante característica de uma estrutura de vizinhança que é ser possível partindo de qualquer solução obter outra qualquer solução por aplicação de uma sequência de operações.

Uma quarta operação, que pode ser incorporada nas heurísticas já referidas (ou noutras (meta-)heurísticas), é a operação de resequenciamento. Nesta operação é resolvido (por um método exacto ou heurístico) o PCV formado pelos vértices de um (sub)circuito o que, mantendo lucro porque os vértices do subconjunto são os mesmos, pode permitir uma redução da distância.

4.3 Programação inteira

Os modelos de programação inteira para os problemas do caixeiro viajante podem ser vistos como extensões aos modelos apresentados na secção anterior.

As variáveis de decisão

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

$$z_i = \begin{cases} 1, se\ vértice\ i\ é\ visitado \\ 0,\ caso\ contrário \end{cases}, \forall i \in N$$

As restrições de grau dos vértices passam a ser

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

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

$$z_1 = 1$$

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

$$z_i \in \{0,1\}, \forall i \in N$$

As restantes restrições e variáveis dos modelos discutidos anterioremente mantêm-se inalteráveis.

Para o PTP, a função objectivo é

$$Min\ z = \sum_{ij \in A} c_{ij} x_{ij} - \lambda \sum_{i \in N} p_i y_i$$

em que λ é uma constante.

Para o OP, a função objectivo é

$$Max\ z = \sum_{i \in N} p_i y_i$$

e adiciona-se a restrição

$$\sum_{ij \in A} c_{ij} x_{ij} \leq c_{max}$$

Para o PCP, a função objectivo é

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

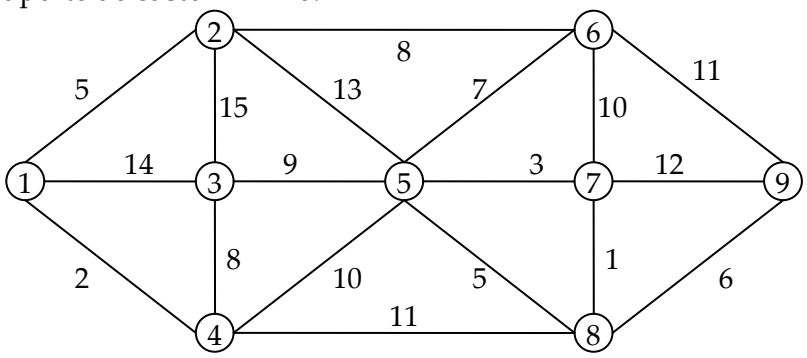
e adiciona-se a restrição

$$\sum_{i \in N} p_i y_i \geq p_{min}$$

5 Exercícios

Exercício 5.1 PCV simétrico: heurísticas construtivas

Para o PCV da figura, aplique as seguintes heurísticas:
a) Vizinho mais próximo partindo de cada um dos vértices;
b) Aresta com menor custo;
c) Inserção do vértice mais próximo partindo do subcircuito 1-2-3-4-1;
d) Clarke e Wright;
e) Árvore de suporte de custo mínimo.



Exercício 5.2 PCV simétrico: heurísticas construtivas

1. Aplique as seguintes heurísticas ao PCV da figura.
a) Vizinho mais próximo partindo do vértice 1;
b) Aresta com menor custo;

2. A dado momento da aplicação da heurística de inserção do vértice mais próximo, o subcircuito actual é 3-4-7-3. Qual o vértice seguinte a inserir nesse subcircuito? Qual a variação de custo do actual para o novo subcircuito?

3. Aplique a heurística de Clarke e Wright partindo dos subcircuitos 1-2-4-5-1, 3-4-7-3 e 4-6-4.

4. A heurística da árvore de suporte de custo mínimo obtém uma solução admissível?

Exercício 5.3 PCV simétrico: heurísticas construtivas e pesquisa local

1. Para o PCV da tabela, aplique as seguintes heurísticas:
a) Vizinho mais próximo partindo do vértice 1;
b) Aresta com menor custo;
c) Inserção do vértice mais próximo partindo do subcircuito 1-2-3-4-1;
d) Clarke e Wright;
e) Árvore de suporte de custo mínimo.

2. Considerando o problema definido pelos 4 primeiros vértices e tendo como solução inicial a solução obtida com a heurística do vizinho mais próximo partindo de 1, obtenha uma solução óptima local para a vizinhança troca de dois vértices.

3, Considerando o problema definido pelos 5 primeiros vértices e tendo como solução inicial a solução obtida com a heurística do vizinho mais próximo partindo de 1, obtenha uma solução óptima local para a vizinhança troca de duas arestas (2-Opt).

	2	3	4	5	6
1	14	17	12	11	19
2	–	20	21	15	13
3	–	–	19	12	11
4	–	–	–	10	18
5	–	–	–	–	19

Exercício 5.4 PCV simétrico: heurísticas construtivas e pesquisa local

Considere a instância do PCV da tabela.

	1	*2*	*3*	*4*	*5*
1	-	3	4	1	9
2	3	-	4	5	3
3	4	4	-	8	2
4	1	5	8	-	9
5	9	3	2	9	-

1. Aplique as seguintes heurísticas:
a) Vizinho mais próximo partindo do vértice 1;
b) Aresta com menor custo;
c) Inserção do vértice mais próximo partindo do subcircuito 1-4-1;
d) Clarke e Wright;
e) Árvore de suporte de custo mínimo

2. Considerando a estrutura de vizinhança troca de duas arestas.
a) As soluções obtidas em 1a) e 1b) são vizinhas?
b) A solução obtida em 1a) é óptima local?
c) A solução obtida em 1b) é óptima local?

Exercício 5.5 PCV Euclidiano: heurísticas construtivas e pesquisa local

Considere um PCV Euclidiano com sete vértices com as coordenadas dadas na tabela.

i	1	2	3	4	5	6	7
x	0	0	1	2	2	3	5
y	0	3	1	0	1	0	5

1. Aplique as seguintes heurísticas:
a) Vizinho mais próximo partindo do vértice 1;
b) Aresta com menor custo;
c) Inserção do vértice mais próximo partindo do subcircuito 1-2-3-4-1;
d) Clarke e Wright;
e) Árvore de suporte de custo mínimo.

2. Mostre que a solução obtida com a heurística do vizinho mais próximo partindo de 1 não é uma solução óptima local para estrutura de vizinhança de troca de dois vértices.

3. Considere o problema com os cinco primeiros vértices da tabela. Através de pesquisa local com a vizinhança de troca de duas arestas (2-Opt), obtenha uma solução óptima local partindo da solução 1-5-4-3-2-1.

Exercício 5.6 PCV Euclidiano: heurísticas construtivas e pesquisa local

Considere um PCV Euclidiano com cinco vértices com as coordenadas dadas na tabela.

i	1	2	3	4	5
x	0	10	7	1	4
y	0	10	2	9	8

1. Aplique as seguintes heurísticas:
a) Vizinho mais próximo partindo do vértice 1;
b) Aresta com menor custo;
c) Inserção do vértice mais próximo partindo do subcircuito 1-2-3-4-1;
d) Clarke e Wright;
e) Árvore de suporte de custo mínimo.

2. Partindo da pior solução obtida com uma das heurísticas da alínea anterior, obtenha uma solução óptima local para as vizinhanças:
a) troca de dois vértices;
b) troca de duas arestas (2-Opt).

Exercício 5.7 PCV assimétrico: heurísticas construtivas

Aplique no problema do caixeiro viajante assimétrico com os custos dados na tabela as heurísticas:

	1	2	3	4	5	6
1	–	14	17	12	11	19
2	10	–	20	21	15	13
3	13	17	–	19	12	11
4	12	13	14	–	10	18
5	18	12	17	26	–	19
6	16	21	22	21	15	–

Aplique as seguintes heurísticas:
a) Vizinho mais próximo partindo de cada um dos vértices;
b) Aresta com menor custo;
c) Inserção do vértice mais próximo partindo do subcircuito 1-2-3-4-1;
d) Clarke e Wright.

Exercício 5.8 Aplicação a sequenciamento

Considere o problema de determinar a ordem pela qual um conjunto de seis tarefas (A, B, C, D, E e F) deve ser executado numa máquina. O tempo de execução de cada tarefa na máquina é independente da ordem pela qual as tarefas são realizadas. No entanto, o tempo de preparação da máquina para a realização de cada tarefa depende da tarefa que foi realizada imediatamente antes.

Na tabela seguinte são dados os tempos de execução e os tempos de preparação (em minutos) de cada tarefa. Por exemplo, a tarefa A demora 3 minutos a ser preparada se a tarefa anterior foi a B, a tarefa A demora 2 minutos a ser preparada se a tarefa anterior foi a C, e assim sucessivamente. Ainda por exemplo, a tarefa A demora 14 minutos a ser executada.

		Tarefa seguinte					
		A	B	C	D	E	F
Tarefa anterior	A	–	7	5	6	4	8
	B	3	–	4	6	7	8
	C	2	7	–	3	7	1
	D	3	2	8	–	2	9
	E	4	4	3	1	–	3
	F	7	4	2	2	4	–
	Tempo de execução	14	25	19	18	13	21

1. Mostre como cada solução deste problema de sequenciamento corresponde a uma solução de um problema do caixeiro viajante, identificando a que correspondem os nodos, os arcos e os custos dos arcos.

2. Obtenha uma solução com cada uma das seguintes heurísticas

a) Vizinho mais próximo partindo de cada um dos vértices;

b) Aresta de menor custo;

c) Inserção do vértice mais próximo partindo do subcircuito 1-2-3-4-1;

d) Clarke e Wright.

Exercício 5.9 *Programação inteira

1. Considere as restrições do modelo Dantzig-Fulkerson-Johnson

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

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

$$\sum_{ij\in A(S)} x_{ij} \leq |S| - 1, \forall S \subset N$$

Indique as restrições que não estão a ser respeitadas na solução $x_{12} = x_{21} = x_{34} = x_{45} = x_{53} = 1$ e todas as restantes variáveis iguais a zero.

2. Indique o valor das variáveis de decisão (x e u) do modelo de Miller-Tucker-Zemlin na solução 13-34-45-52-21 de um PCV com cinco nodos.

Exercício 5.10 *Programação inteira

Considere o PCV da tabela.

	1	2	3	4	5
1	–	4	7	2	1
2	3	–	3	1	12
3	4	5	–	2	6
4	5	1	6	–	10
5	6	9	2	5	–

1. Apresente os seguintes modelos:
a) Dantzig-Fulkerson-Johnson;
b) Miller-Tucker-Zemlin;
c) fluxos (agregado);
d) fluxos (desagregado);
e) etapas.

2. Utilizando um *solver* de programação inteira, obtenha uma solução óptima com base em cada um dos modelos da alínea anterior.

Exercício 5.11 *GRASP

Aplique a meta-heurística GRASP no PCV definido pela tabela para valores de α de 0.5 e 1. Na componente aleatória do algoritmo, tome decisões arbitrárias. Utilize a heurística construtiva do vizinho mais próximo, seleccionando, em cada iteração, arbitrariamente o vértice inicial, e utilize a heurística de troca de duas arestas para a pesquisa local. Use como critério de paragem não ter havido melhoria da solução incumbente.

	2	3	4	5
1	4	7	2	1
2	–	3	1	12
3	–	–	2	6
4	–	–	–	10

6 Bibliografia

- J. E. Beasley , OR-Library, http://people.brunel.ac.uk/~mastjjb/jeb/info.html
- D. Feillet, P. Dejax, M. Gendreau. "Traveling salesman problems with profits." Transportation science 39.2 (2005): 188-205.
- D. S. Johnson, L. A. McGeoch. "The traveling salesman problem: A case study in local optimization." Local search in combinatorial optimization 1 (1997): 215-310.
- S. Mertens, TSP Algorithms in Action Animated Examples of Heuristic Algorithms, http://wase.urz.uni-magdeburg.de/mertens/TSP/
- J. F. Oliveira, M. A. Carravilla, Heuristics and Local Search, http://paginas.fe.up.pt/~mac/ensino/docs/OR/HowToSolveIt/ConstructiveHeuristicsForTheTSP.pdf, 2001
- A. J. Orman, H.P. Williams, A Survey of Different Integer Programming Formulations of the Travelling Salesman Problem, Optimisation, Econometric and Financial Analysis, Advances in Computational Management Science 9, pp 91-104, 2007
- G. Reinelt, The Traveling Salesman Computational Solutions for TSP Applications, Lecture Notes in Computer Science 840, online edition, 2001.
- TSP - Principal sítio sobre o PCV que inclui muita informação, muitas aplicações e permite download do software concorde http://www.math.uwaterloo.ca/tsp/index.html

7 Resultados de aprendizagem

- Aplicar heurísticas construtivas para resolver instâncias de pequena dimensão do problema do caixeiro viajante.
- Aplicar heurísticas de pesquisa local para resolver instâncias de pequena dimensão do problema do caixeiro viajante.
- *Formular o problema do caixeiro viajante (e variantes com lucros) através de modelos de programação inteira.
- *Resolver problema do caixeiro viajante (e variantes com lucros) com software, nomeadamente o Solver e openSolver para Excel e o IBM ILOG CPLEX Optimization Studio.