Grafo Euleriano

Um caminho simples ou um circuito simples é dito *euleriano* se ele contém todas as arestas de um grafo. Um grafo que contém um circuito euleriano é um *grafo euleriano*. Um grafo que não contém um circuito euleriano, mas contém um caminho euleriano será chamado *grafo semi-euleriano*. As figuras 1a e 1b ilustram grafos euleriano e semi-euleriano, respectivamente.

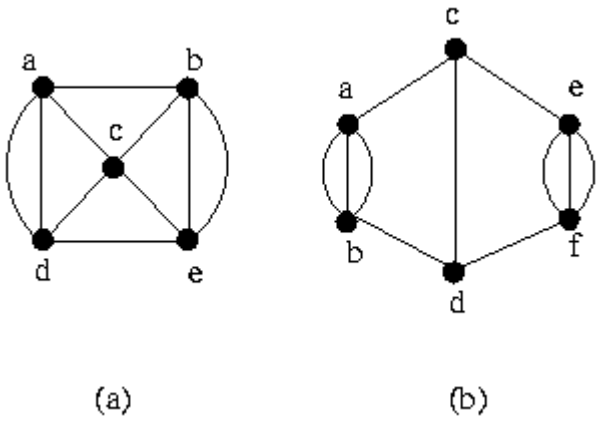


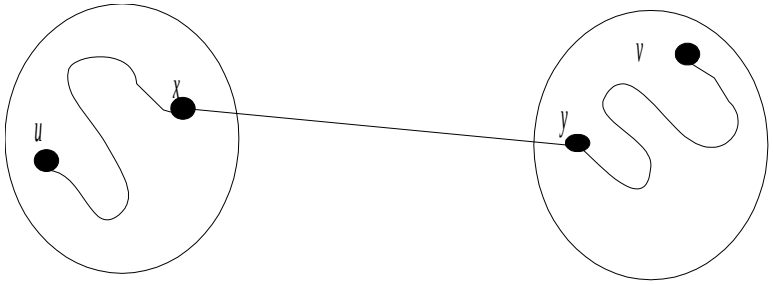
Figura 1

Por definição, um caminho é sempre conexo. Como um circuito euleriano contém todas as arestas de um grafo, um grafo euleriano é sempre conexo, com a exceção dos vértices isolados.

Lema 1: *Dado um grafo conexo com todos os vértices de grau par, então qualquer par de vértice faz parte de um caminho simples e fechado.*

Prova: (Por contradição). Suponha que exista um par de vértices *u* e *v* que não admitem um circuito simples em comum. Como o grafo é conexo, então existe um caminho que liga *u* e

v. Portanto, existe uma aresta (*x*,*y*) cuja a remoção desconexa o grafo, caso contrário haveria um outro caminho disjunto alternativo ligando *u* e *v*.



Portanto, a remoção da aresta (*x*,*y*) desconexa grafo gerando duas componentes conexas, sendo o vértice x e y pertencentes a componentes diferentes. Com isso, ambos os vértices passarão a ser os únicos vértices de grau ímpar. Isso contradiz o fato que o número de vértices de grau ímpar de um subgrafo deva ser par. C.Q.D.

Teorema 1: *Um grafo conexo G é um grafo euleriano se e somente se todo vértice de G possui grau par*.

Prova:

Ida: Seja G um grafo euleriano. Logo ele contém um circuito euleriano. Por cada ocorrência de vértice desse caminho, existe uma aresta que chega nesse vértice e outra que sai desse vértice. Como toda aresta faz parte do caminho, isto é, nenhuma aresta fica fora do caminho, necessariamente o número de arestas incidentes em cada vértice é par.

Volta: Como todo vértice possui grau par, então na construção de um caminho sempre é possível chegar e sair de um vértice por arestas diferentes ainda não utilizada. Assim, é possível sair de vértice *v* e retornar a ele sem repetição de arestas (Lema 1). Seja C_1 um circuito contendo *v* construído de maneira arbitrária. Logo, se C_1 contém todas as arestas de G, temos um circuito euleriano. Senão, retiramos de G todas as arestas que fazem parte de C_1. No grafo resultante G', todos os vértices também possuem grau par e necessariamente um deles faz parte de C_1, senão o grafo não seria conexo. Recomecemos o mesmo processo com o grafo G', partindo de um vértice comum com C_1, obtendo assim um novo circuito C_2. A figura 2 mostra que dois circuitos que têm um vértice em comum podem formar um circuito único: chegando no vértice comum em um dos dois circuitos, continuamos o percurso no outro circuito. Continuando esse processo, necessariamente obteremos um circuito único que contém todas as arestas de *G*. Logo o grafo é euleriano. C.Q.D.

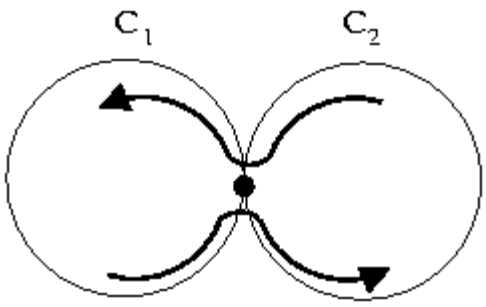


Figura 2

Um aspecto muito interessante dessa prova é que ela sugere um algoritmo para identificar um circuito euleriano. Considere por exemplo o grafo ilustrado na figura 3a.

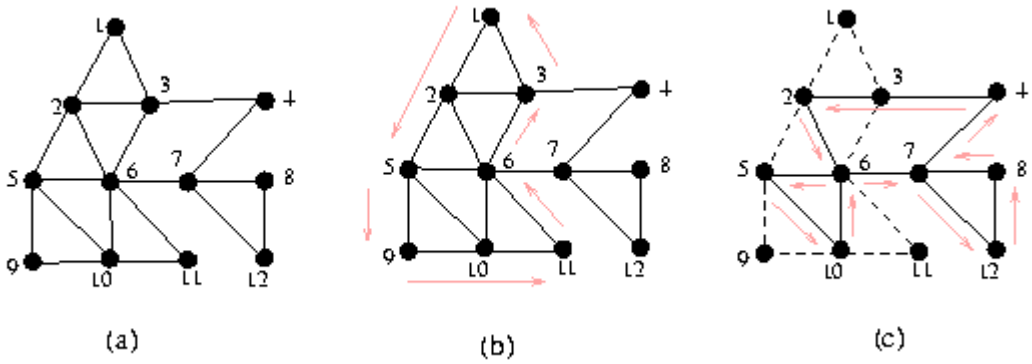


Figura 3

Supondo que começamos pelo vértice 1, e escolhemos aleatoriamente um aresta nunca visitada a cada vértice visitado, até voltar ao vértice 1 sem poder sair mais. A figura 3b mostra um circuito obtido, que consiste na sequência 1, 2, 5, 9, 10, 11, 6, 3 e 1. Como sobram arestas não percorridas, devemos recomeçar a partir de um vértice desse circuito. Supondo que o vértice 6 foi escolhido, podemos obter, como ilustrado na figura 3c, o circuito 6, 7, 12, 8, 7, 4, 3, 2, 6, 5, 10 , 6. Combinando esses circuito com o que já tinhamos, obtemos um novo circuito 1, 2, 5, 9, 10, 11, 6, 7, 12, 8, 7, 4, 3, 2, 6, 5, 10, 6, 3, 1. Como esse circuito cobre o grafo inteiro, não é preciso recomeçar o processo: já temos o nosso circuito euleriano.

Esse algoritmo é conhecido como o algoritmo de Hierholzer. Suponhamos que um caminho de um vértice v_1 até v_k é representado por uma lista $[v_1, a_1,..., a_{k-1}, v_k]$, que alterna vértices e arestas. Eis uma descrição do algoritmo de Hierholzer (supondo que já sabemos que o grafo é euleriano):

função *Hierholzer*(G = (V,E): *grafo*) : *caminho*

G' := G { G' = (V', E')}
v_0 := um vértice de G'
C := [v_0] {Inicialmente, o circuito contém só v_0}
Enquanto E' não vazio

v_i := um vértice de C tal que d(v_i) > 0 em G'
C' := Circuito em G' que contém v_i
G' := G' - {a | a é aresta contida em C'}
Em C, substituir o vértice v_i pelo circuito C'

Retornar C

Complexidade do Algoritmo de Hierholzer

(Exercício em sala de aula)

O Problema do Carteiro Chinês

As aplicações dos caminhos eulerianos se relacionam, basicamente, com problemas de atendimento seqüencial, sobre um conjunto de usuários de um serviço oferecido no interior de uma rede de tráfego, tais como, entrega de correio, coleta de lixo, vendas por atacado, etc.

Um problema interessante ligado ao conceito de grafo semi-euleriano é o Problema do Carteiro Chinês. Imagine um carteiro que deve percorrer um roteiro todo dia. O problema é de identificar esse roteiro de maneira a minimizar a distância total percorrida. Essa situação pode ser representada por um grafo onde as arestas correspondem às ruas e os vértices correspondem aos cruzamentos.

Se o grafo é euleriano, a solução consiste simplesmente em achar um circuito euleriano. Mais interessante é o caso de um grafo não euleriano. Consideremos por exemplo um grafo semi-euleriano, como ilustrado na figura 4. Supondo que o carteiro quer voltar ao lugar de origem, portanto com certeza para cada um dos vértices 1 e 8, uma das arestas adjacentes será atravessada no mínimo duas vezes.

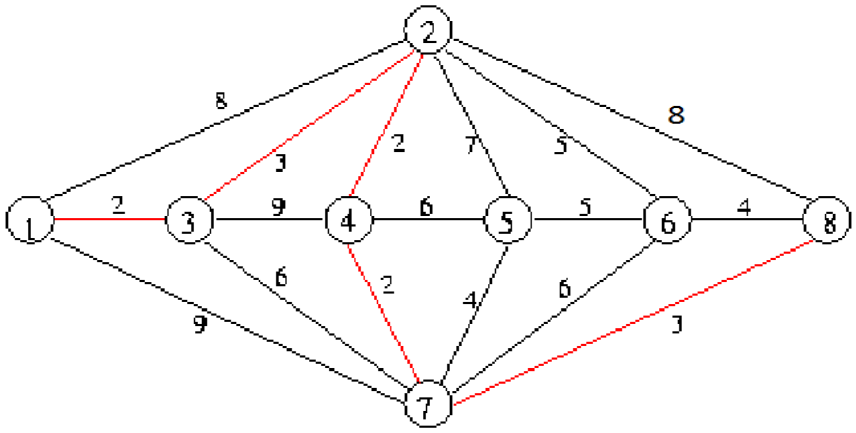


Figura 4

Uma solução ao problema é a seguinte. Transforme o grafo em grafo euleriano, duplicando as arestas que formam o caminho mais curto entre os dois vértices de grau ímpar. Esse caminho é indicado em vermelho na figura 4. O grafo assim obtido é euleriano. Agora podemos aplicar um dos algoritmos de identificação de circuito euleriano.

Portanto, dado um grafo $G=(V,E)$ conexo com custo nas arestas, o objetivo do ***Problema do Carteiro Chinês*** é encontrar um caminho fechado de custo mínimo passando por cada aresta pelo menos uma vez.

Solução 1: Se o grafo for de Euler, então o caminho pode ser encontrado, e conseqüentemente, o seu respectivo custo, através do algoritmo de Hierholzer. O custo é dado pela soma dos custos de todas as arestas do grafo.

Solução 2: Se o grafo não for de Euler, então algumas arestas terão que ser repetidas. A maneira clássica de resolver este problema é acrescentando arestas artificiais ao grafo original de forma a obter um novo grafo $G'=(V,E')$. Isto deve ser feito de maneira que as arestas artificiais acrescentadas transformem todos os vértices de grau ímpar de G, em vértices de grau par. As arestas artificiais correspondem aos eventuais percursos repetidos de custo mínimo entre pares de vértices de grau impar.

Algoritmo para o Problema do Carteiro chinês

P1. Identifique os vértices de grau ímpar. Digamos que existam α vértices de grau ímpar.

P2. Determine as combinações possíveis de vértices de grau ímpar, interligando-os com arestas artificiais, formando grafos expandidos, contendo somente vértices de graus par.

P3. Selecione o grafo $G=(V,E)$ expandido que apresentar a menor extensão total das arestas artificiais.

P4. Determine um roteiro de Euler par o grafo $G=(V, E)$.

Observe que as possíveis combinações de vértices de grau ímpares podem ser um número muito grande. Supondo que o grafo tenha 2*k* vértices de grau ímpar, em geral, o número *r* de percursos é dado por:

$$r = \frac{2k!}{2^k . k!} = (2k-1)(2k-3)...3.1$$

Para ter uma idéia, observe estes exemplos:

# de vértices = 2k	4	6	8	10	20
# de combinações	3	15	105	945	$655x10^6$

A enumeração total torna-se rapidamente inviável.

Para problemas pequenos, ou seja, número pequeno de vértices de grau ímpar, por inspeção do grafo é geralmente fácil achar a combinação de custo mínimo, ou seja, a introdução de arestas artificiais.

Exemplo:
Um caminhão de coleta de lixo domiciliar deve percorrer as ruas de uma determinada zona urbana representada pelo grafo abaixo, partindo de ponto A e retornando a este ponto.
Determinar:

a) o roteiro que torne mínima a distância percorrida pelo veículo;
b) a extensão total do percurso dentro da zona.

Caminho:
Custo:

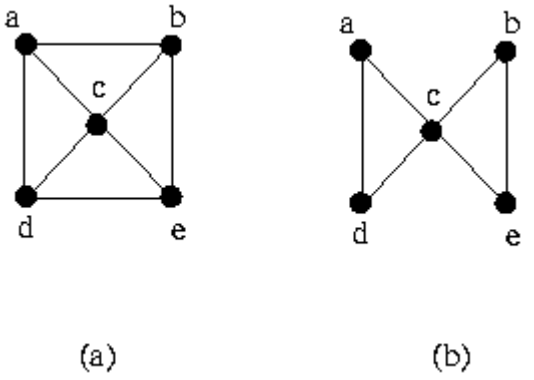
Algoritmos de Emparelhamento.

Considere um grafo completo K_α, sendo α o número de vértices de grau ímpar encontrados no grafo original. Cada aresta do grafo K_α deve representar um caminho de custo mínimo entre os vértices de grau ímpar do grafo original. Portanto, encontrar a melhor combinação de pares de vértices (arestas artificiais) significa encontrar um emparelhamento de custo mínimo com $\alpha/2$ arestas em K_α. Este problema pode facilmente ser transformado em um problema de emparelhamento de peso máximo (*Maximum Weight Matching*). Para isto, na literatura podem ser encontrados algoritmos de complexidade polinomial. Porém, a implementação destes algoritmos não é trivial. Sendo assim, não trataremos sobre isso neste curso. Aos interessados, sugere-se uma bibliografia especializada, ou até mesmo na internet é possível encontrar exemplares de algoritmos para este problema.

Problemas Hamiltonianos

Definição: Um **circuito hamiltoniano** em um grafo conexo *G* é definido como um caminho elementar, fechado passando em cada vértice de *G* exatamente uma vez. Um grafo que admite um circuito hamiltoniano é um **grafo hamiltoniano**.

Evidentemente nem todo grafo é hamiltoniano. A figura abaixo ilustra dois grafos, o primeiro é hamiltoniano (a) e o segundo não (b).



Portanto, um circuito hamiltoniano de um grafo de *n* vértices consiste de exatamente *n* arestas. Um circuito hamiltoniano equivale a uma permutação dos vértices, assim, o número máximo de caminhos hamiltonianos de um grafo com n vértices é igual a *n*!.

Os problemas desta classe são, de modo geral, de dificuldade maior que os problemas eulerianos. Há um grande número de resultados úteis, ao lado de muitas questões em aberto; por outro lado os algoritmos para descobrir um circuito hamiltoniano são em geral, muito mais trabalhosos. Enquanto que um grafo euleriano pode ser verificado em tempo linear, não se conhece algoritmo polinomial para verificar se um grafo é hamiltoniando, exceto para casos particulares. Por exemplo, não é complicado verificar que um grafo completo possui um circuito hamiltoniano em tempo $O(n^2)$.

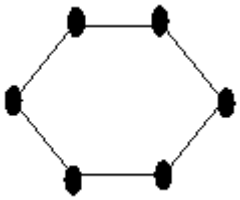
Propriedades para grafos hamiltonianos

Existem muitos resultados para acerca de grafos hamiltoniados, porém, há poucos teoremas relativamente úteis. A seguir são apresentados dois teoremas.

Teorema de Ore. Uma condição suficiente (mas não necessária) para que um grafo G seja hamiltoniano é que a soma dos graus de cada par de vértices não adjacentes seja no mínimo n.

Teorema de Dirac: Uma condição suficiente (mas não necessária) para que um grafo simples G possua um ciclo hamiltoniano, é que o grau de cada vértice em G seja pelo menos igual a n/2, onde n é o número de vértices em G.

Observe que estes teoremas não são suficientes para determinar se um grafo é hamiltoniano. Observe o grafo abaixo que é um grafo hamiltoniano que não obedece as condições dos teoremas acima. O grau de cada vértice é 2, que é menos que 6/2=3 e, além disso, a soma dos graus de qualquer par de vértices não adjacentes é sempre 4, que é menor que n=6 (vértices).



A seguir é apresentado um algoritmo que determina os circuitos hamiltoniados.

Método Exato: Método "Composição Latina"

Trata-se de um método algébrico para enumeração de caminhos hamiltonianos. Este método gera todos os caminhos simples por multiplicação de matrizes.

Considere a matriz $B(nxn)$ da seguinte forma:

$$B_{ij} = \begin{cases} v_j, \text{se existe a aresta } (v_i, v_j) \\ 0, \text{caso contrário} \end{cases}$$

Passos do Algoritmo

P1. Construa a matriz B e a matriz de adjacência A do grafo dado.

P2. Faça $P_1 \leftarrow A$;

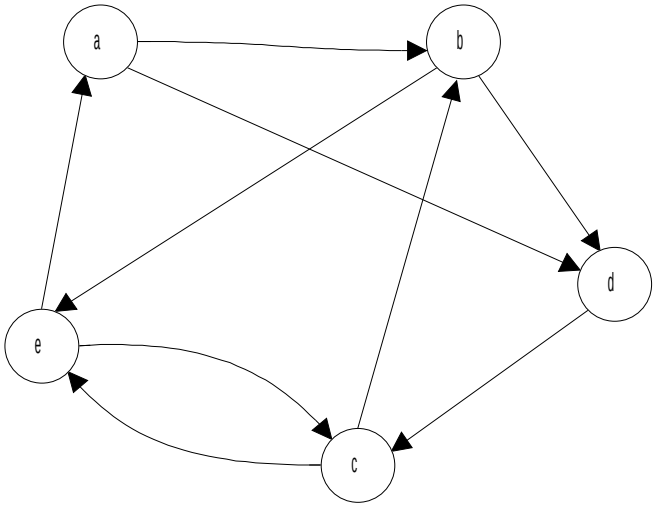
P3. Para i = 1,2,...., n-2 faça

$P_{i+1} \leftarrow B$ x P_i

Obs:

1) Os elementos da matriz P são cadeias de caracteres (vértices) e não números. A operação *multiplicação* significa a concatenação do caracteres e a *soma* representa a divisão de duas ou mais cadeias de caracteres.
2) Cada elemento da matriz P_i representa um caminho hamiltoniano de comprimento i + 1 entre os vértices **s** e **t.**
3) Para todo P_{i+1} a diagonal é zero, assim como todo caminho de **s** até **t** contendo **s**.

Exemplo:
Determinar as caminhos hamiltonianos do grafo abaixo:



Solução pelo método da composição latina.

Passo 1: $A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 0 & b & 0 & d & 0 \\ 0 & 0 & 0 & d & e \\ 0 & b & 0 & 0 & e \\ 0 & 0 & c & 0 & 0 \\ a & 0 & c & 0 & 0 \end{bmatrix}$

Passo 2: $P_1 = A$

Passo 3: $P_2 = B$xP_1; $P_3 = B$xP_2; $P_4 = B$xP_3

Eliminam-se nas matrizes P_i os termos sublinhados, pois são caminhos de ***s*** até ***t*** que contem ***s***. Faz-se também a diagonal igual a zero.

$$P_2 = \begin{bmatrix} 0 & 0 & d & b & b \\ e & 0 & d+e & 0 & 0 \\ e & 0 & \underline{e} & b & b \\ 0 & c & 0 & 0 & c \\ 0 & a+c & 0 & a & \underline{c} \end{bmatrix} = \begin{bmatrix} 0 & 0 & d & b & b \\ e & 0 & d+e & 0 & 0 \\ e & 0 & 0 & b & b \\ 0 & c & 0 & 0 & c \\ 0 & a+c & 0 & a & 0 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} \underline{be} & dc & bd+be & 0 & dc \\ 0 & \underline{bc+ea+ec} & 0 & ea & dc \\ be & ea+\underline{ec} & \underline{bd+be} & ea & 0 \\ ce & 0 & 0 & \underline{cb} & cb \\ ce & 0 & ad & ab+cb & \underline{ab+cb} \end{bmatrix}$$

$$P_4 = \begin{bmatrix} \underline{dce} & 0 & 0 & \underline{bea} & bdc+dcb \\ dce & 0 & ead & \underline{eab}+\underline{ecb} & \underline{dcb} \\ 0 & 0 & \underline{ead} & bea+eab+\underline{ecb} & \underline{bdc}+eab+\underline{ecb} \\ cbe & cea & 0 & \underline{cea} & 0 \\ \underline{cbe} & adc+\underline{cea} & abd+\underline{abe} & \underline{cea} & \underline{abc} \end{bmatrix}$$

Os caminhos hamiltonianos são: abdce, adcbe, bdcea, baedc, cbead, ceabd, dcbea, dceab, eadcb, eabdc.

Se as arestas são valoradas, então pode-se determinar o caminho de menor custo.

Os ciclos disjuntos contendo todos os vértices são: **abdcea , adcbea**

Obs: A maioria dos algoritmos exatos são normalmente da classe ”*Branch – and- Bound*”. Trata-se de buscas em árvores, caracterizadas pelo particionamento do conjunto de soluções por um critério dado. Em problemas práticos estes algoritmos são inviáveis, pois, o tempo de computação cresce rapidamente com a dimensão do problema.

Problema do Caixeiro Viajante (PCV)

(*Traveling Salesman Problem*)

É um problema de grande importância prática que consiste na determinação da rota de menor custo para uma pessoa que parta de uma cidade e deva visitar diversas outras, passando pelo menos uma vez em cada cidade e retornando ao ponto de partida.

O custo pode ser medido em termos de dinheiro, tempo, distância, etc, e as opções existentes para as diferentes etapas de viagens correspondem as arestas de um grafo valorado.

Se o grafo não for muito grande, e especialmente, se não tiver muitos arcos, será possível enumerar os circuitos hamiltonianos e depois achar o valor de cada um deles; mas no caso geral, isso é impossível na prática.

Algoritmos Heurísticos para o PCV

Os métodos exatos para resolver o PCV não são computacionalmente eficientes, o que tem levado os pesquisadores a utilizarem algorítimos heurísticos, que geram soluções satisfatórias, com consumo de tempo computacional muitas vezes menor do que um algoritmo exato.

Uma heurística é um processo que procura boas soluções de um problema a um custo computacional razoável, porém, sem ser capaz de garantir a otimalidade ou até, em muitos casos, de estabelecer quão perto uma dada solução viável está da solução ótima. Portanto, não existe prova de corretude desses algoritmos heurísticos.

Os algoritmos heurísticos encontrados na literatura podem ser divididos nas seguintes classes:

- Heurística de Construção;
- Heurística de Melhoramento;
- Heurística mista.

Heurística de Construção

As heurísticas de construção mais conhecidas são:

- Vizinho mais próximo;
- Inserção mais próxima;
- Inserção mais distante;
- Inserção mais barata;
- Algoritmos das Economias (Clark-Wright).

Esses algoritmos serão vistos em sala de aula.

Heurística de Melhoramento

As heurísticas de melhoramento (ou de melhoria iterativa) mais conhecidas são:

- K-opt ou K-melhoramento; (em sala de aula)
- *Simulated Annealing;*
- Algoritmos Genéticos;
- Busca Tabu;
- etc.

Neste curso será visto apenas o K-Opt, porém, na literatura podem ser encontrados outros algoritmos heurísticos como citados acima.

Heurística de Melhoramento K-Opt

Trata-se de uma estratégia de troca de arestas que podem ser usadas para melhorar uma solução obtida por algum algoritmo de construção. A heurística K-Opt elimina K arestas não adjacentes do roteiro do PCV e reconectam esses k caminhos em ordem diferentes através de outras k arestas. Entretanto, as experiências computacionais com K=2 e K=3 têm apresentado os melhores resultados.

A Heurística 2-OPT

A heurística 2-opt é uma das técnicas mais conhecida para melhorar uma solução para o problema do caixeiro viajante. Inicia-se com um ciclo *H*; retira-se 2 arestas de *H*, assim produzindo 2 grafos desconectados. Os dois grafos são reconectados de tal maneira que produza outro ciclo *H'*. Assim *H* e *H'* diferem em exatamente 2 arestas. Calcula-se o custo *w*(*H'*) do ciclo *H'*. Se *w*(*H'*)< *w*(*H*), o ciclo *H* é substituído por *H'* e repete-se o processo; caso contrário, um outro par de arestas é trocado. As mudanças continuam até que nenhum melhoramento poder se feito trocando 2 arestas. A solução final que não pode ser melhorada mudando apenas 2 arestas é chamada de uma solução 2-opt (Figura 1).

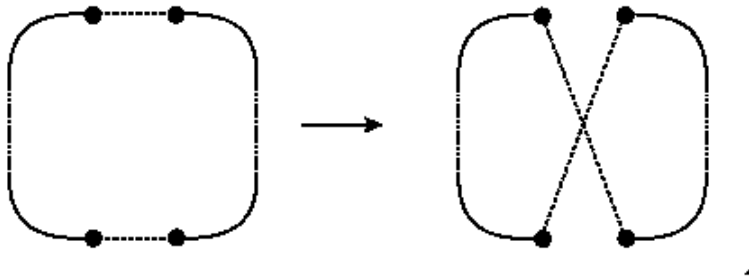


Figura 1. Ilustração da heurística 2-Opt

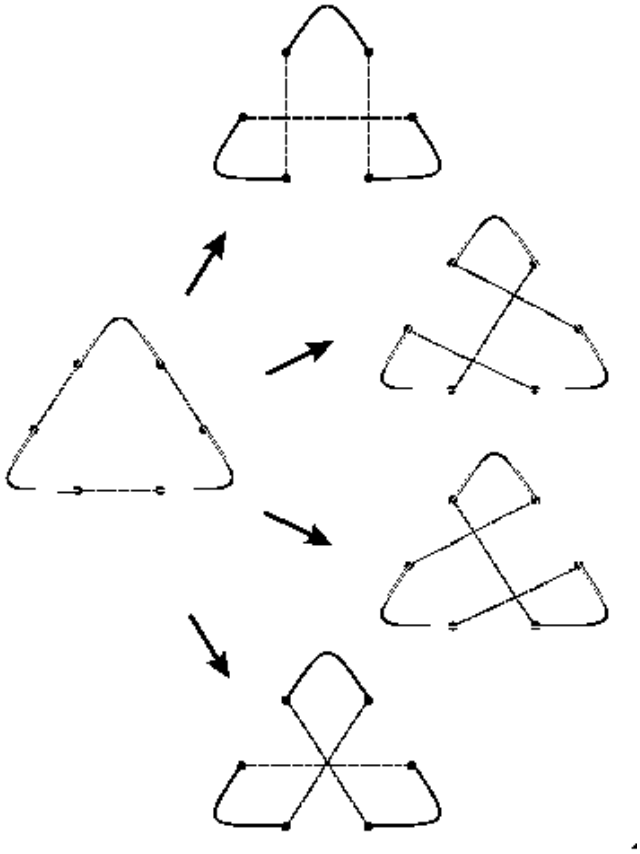


Figura 2. Ilustração da heurística 3-Opt.

De maneira análoga a heurística 2-Opt pode se criar a heurística 3-Opt. Neste caso, a mudança de 3 arestas podem ser combinadas em 4 formas diferente, conforme a Figura 2.

Formalização da Heurística 2-Opt

Dado um ciclo inicial contendo o seguinte conjunto de arestas $H=\{x_1, x_2, ..., x_n\}$ na ordem $x_1, x_2, ..., x_n$. Define-se $w(H)$ o custo associado ao ciclo H. Seja $X=\{x_i,x_j\}$ um conjunto de 2 arestas de H é apagado e substituído por outro conjunto de arestas $Y=\{y_p,y_q\}$, obtendo-se um novo ciclo $H'= (H - X) \cup Y$. Se $w(H')<w(H)$ então H é substituído por H'.

Observe que:

1) as duas arestas x_i, x_j de X não podem ser adjacentes;
2) uma vez que X tenha sido escolhido, o conjunto Y é determinado.

Assim, é possível gerar $\frac{n(n-3)}{2}$ ciclos H' a partir de um dado ciclo H.

Denota-se δ como sendo o melhoramento

$$\delta = w(H) - w(H') = (w(x_i) + w(x_j)) - (w(y_p) + w(y_q)).$$

O algoritmo examina todos os ciclos *H'* a fim de obter um ciclo com o máximo valor de δ. O valor máximo é denotado por δ_{max}.

A seguir é descrito o algoritmo deste método para alcançar uma solução 2-opt para o problema do caixeiro viajante através de sucessivas trocas de 2 arestas (Syslo *et al*, 1983)

Procedure 2-OPT

```
begin
    < seja H= (x1, x2, ..., xn) o ciclo corrente >;
    repeat
        δmax :=0;
        for i:=1 to (n-2) do
            for j:=(i+1) to n  < ou (n-1) quando i=1 > do
                if (w(xi) + w(xj) - w(yp) - w(yq))> δmax then
                    begin
                        δmax:= (w(xi) + w(xj) - w(yp) - w(yq));
                        < guardar i e j >;
                    end;
        if  δmax > 0 then H:= H- {xi , xj}  {yp ,yq};
    until δmax=0;
end;
```

Exercícios.

Exercício 1. Faça a análise de complexidade para cada algoritmo heurístico construtivo apresentado.

Exercício 2. Faça a análise de complexidade para os algoritmos heurísticos melhorativo 2-Opt e 3-Opt.

Exercício 3. "Se um digrafo é euleriano, ele é fortemente conexo". Esta afirmativa está correta? Por quê? A recíproca é verdadeira? Por quê?