Problema do Caixeiro Viajante

Marina Andretta

ICMC-USP

2 de março de 2019

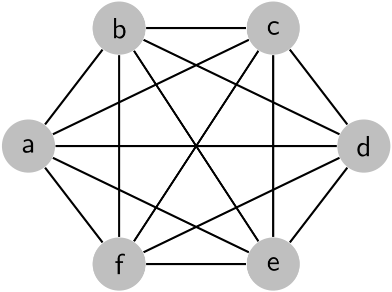
Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (ou *Travelling Salesperson Problem*, TSP) pode ser formulado da seguinte maneira.

Dado um grafo $G = (V, A)$ não-orientado, com custos c associados às arestas, encontrar um ciclo hamiltoniano (ou seja, um ciclo - ou circuito - que passa por todos os vértices de G exatamente uma vez) de custo mínimo.

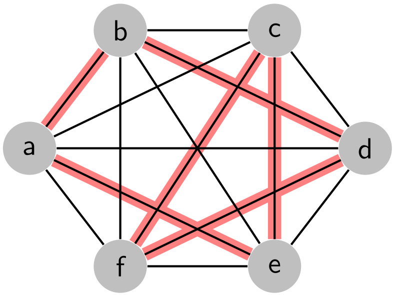
Sem perda de generalidade, podemos definir $V = \{v_1, v_2, ..., v_n\}$, e custos c_{ij} para cada aresta (v_i, v_j) em A. Vamos supor também que o grafo G é completo, ou seja, $(v_i, v_j) \in A$ para todo $i \neq j$. Caso ele não seja, podemos inserir em A as arestas que não existem, atribuindo um custo muito alto a elas.

Exemplo



	a	b	c	d	e	f
a	-	1	2	1	1	2
b	1	-	7	1	4	3
c	2	7	-	3	1	1
d	1	1	3	-	8	1
e	1	4	1	8	-	1
f	2	3	1	1	1	-

Exemplo



	a	b	c	d	e	f
a	-	1	2	1	1	2
b	1	-	7	1	4	3
c	2	7	-	3	1	1
d	1	1	3	-	8	1
e	1	4	1	8	-	1
f	2	3	1	1	1	-

O ciclo ótimo é dado por (a, b, d, f, c, e, a), com custo 6.

Neste caso, é fácil ver que ele é ótimo, porque o custo de todas arestas é maior ou igual a 1 e há 6 arestas no ciclo.

Aplicações do Problema do Caixeiro Viajante

Uma aplicação do Problema Viajante que não é muito óbvia foi a feita por um grupo da AT&T para calcular sequências de DNA.

No caso deles, uma coleção de sequências de DNA, cada uma de tamanho k, deveriam ocorrer em uma sequência universal. Ou seja, cada uma das sequências deve ser uma sub-sequência da sequência universal.

O objetivo é determinar uma sequência universal com menor tamanho possível.

Na modelagem deles, os vértices do Problema do Caixeiro Viajante são definidos como as sequências de DNA.

Todos os vértices são conectados por arestas.

O custo de uma aresta uv é dado por k menos o tamanho da maior sobreposição entre as sequências u e v.

Modelagem do Problema do Caixeiro Viajante como problema de otimização

Exercício: reúna um grupo de até 4 pessoas e desenvolva um modelo de otimização para o Problema do Caixeiro Viajante.

Modelagem do Problema do Caixeiro Viajante como problema de otimização

Uma forma de modelar o Problema do Caixeiro Viajante como um problema de otimização é a dada a seguir.

Defina uma variável x_{ij} para cada aresta (v_i, v_j) em A, ou seja, para $i = 1, ..., n$ e $j = 1, ..., n$, com $i \neq j$.

A variável x_{ij} terá valor 1 caso v_i e v_j apareçam nesta ordem no ciclo escolhido (ou seja, no ciclo escolhido, vai-se do vértice v_i para o vértice v_j).

Caso contrário, x_{ij} terá valor 0.

Função objetivo

Com essas variáveis definidas, o objetivo passa a ser minimizar a função

$$\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} c_{ij} x_{ij}.$$

Restrições

Agora vamos definir as restrições do modelo.

Como queremos encontrar um ciclo que passa por todos os vértices exatamente uma vez, vamos impor que haja exatamente uma aresta "entrando" em v_i e uma "saindo" de v_i para cada vértice v_i de G.

Restrições

Agora vamos definir as restrições do modelo.

Como queremos encontrar um ciclo que passa por todos os vértices exatamente uma vez, vamos impor que haja exatamente uma aresta "entrando" em v_i e uma "saindo" de v_i para cada vértice v_i de G.

Isso pode ser feito com as restrições

$$\sum_{j=1, j \neq i}^{n} x_{ji} = 1 \text{ e } \sum_{j=1, j \neq i}^{n} x_{ij} = 1,$$

para cada vértice v_i.

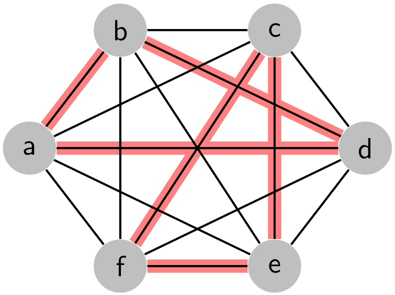
Sub-ciclo

Essas restrições são suficientes?

Sub-ciclo

Essas restrições são suficientes?

Veja este exemplo:



	a	b	c	d	e	f
a	-	1	2	1	1	2
b	1	-	7	1	4	3
c	2	7	-	3	1	1
d	1	1	3	-	8	1
e	1	4	1	8	-	1
f	2	3	1	1	1	-

Sub-ciclo

Neste caso, temos dois ciclos: (a, b, d, a) e (f, c, e, f), ambos com custo 3. Ou seja, o custo total é 6 (que já sabemos que é ótimo).

Note que as restrições são todas respeitadas, porque, para cada vértice, há exatamente uma aresta "entrando" e outra "saindo" dele no ciclo.

Precisamos, então, de restrições para eliminar esses sub-ciclos.

Restrições

Uma alternativa é definir $\mathcal{S}$ o conjunto de todos os subconjuntos V com número de elementos entre 2 e $n-2$. Assim, para cada $S \in \mathcal{S}$, podemos impor que haja pelo menos uma aresta do ciclo escolhido que "sai" deste conjunto S.

Restrições

Uma alternativa é definir $\mathcal{S}$ o conjunto de todos os subconjuntos V com número de elementos entre 2 e $n-2$. Assim, para cada $S \in \mathcal{S}$, podemos impor que haja pelo menos uma aresta do ciclo escolhido que "sai" deste conjunto S.

Esta restrição pode ser escrita da seguinte forma

$$\sum_{i \in S, j \notin S} x_{ij} \geq 1,$$

para todo $S \in \mathcal{S}$.

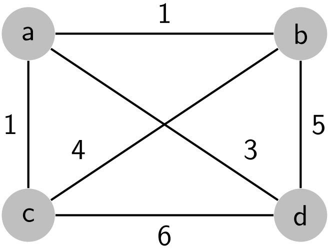
Modelo completo

Assim, nosso modelo fica:

$$\begin{array}{lll}
\text{Minimizar} & \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} c_{ij} x_{ij} & \\
\text{sujeita a} & \sum_{j=1, j \neq i}^{n} x_{ji} = 1, & i = 1, ..., n, \\
& \sum_{j=1, j \neq i}^{n} x_{ij} = 1, & i = 1, ..., n, \\
& \sum_{i \in S, j \notin S} x_{ij} \geq 1, & \forall S \in \mathcal{S}, \\
& x_{ij} \in \{0, 1\}, & i = 1, ..., n, j = 1, ..., n, i \neq j.
\end{array}$$

Outro exemplo

Considere o seguinte exemplo:



Neste caso, $\mathcal{S} = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}$.

Modelo para o exemplo

$$\begin{array}{ll}
\text{Minimizar} & x_{ab} + x_{ba} + x_{ac} + x_{ca} + 3x_{ad} + 3x_{da} + 4x_{bc} + 4x_{cb} + \\
 & 5x_{bd} + 5x_{db} + 6x_{cd} + 6x_{dc} \\
\text{sujeita a} & x_{ba} + x_{ca} + x_{da} = 1, \\
 & x_{ab} + x_{ac} + x_{ad} = 1, \\
 & x_{ab} + x_{cb} + x_{db} = 1, \\
 & x_{ba} + x_{bc} + x_{bd} = 1, \\
 & x_{ac} + x_{bc} + x_{dc} = 1, \\
 & x_{ca} + x_{cb} + x_{cd} = 1, \\
 & x_{ad} + x_{bd} + x_{cd} = 1, \\
 & x_{da} + x_{db} + x_{dc} = 1,
\end{array}$$

Continuação do modelo para o exemplo

$$
\begin{aligned}
&x_{ac} + x_{ad} + x_{bc} + x_{bd} \geq 1,\\
&x_{ab} + x_{ad} + x_{cb} + x_{cd} \geq 1,\\
&x_{ab} + x_{ac} + x_{db} + x_{dc} \geq 1,\\
&x_{ba} + x_{bd} + x_{ca} + x_{cd} \geq 1,\\
&x_{ba} + x_{bc} + x_{da} + x_{dc} \geq 1,\\
&x_{ca} + x_{cb} + x_{da} + x_{db} \geq 1,\\
&x_{ab}, x_{ba}, x_{ac}, x_{ca}, x_{ad}, x_{da}, x_{bc}, x_{cb}, x_{bd}, x_{db}, x_{cd}, x_{dc} \in \{0, 1\}.
\end{aligned}
$$

Modelagem do Problema do Caixeiro Viajante como problema de otimização

Um problema desta modelagem é que a quantidade de restrições cresce exponencialmente com o número de vértices de G, por causa da restrição de eliminação de sub-ciclos.

Existem outras modelagens conhecidas que tentam evitar que este tipo de coisa aconteça.

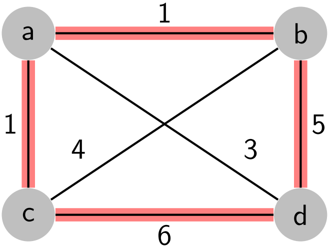
Algoritmos para resolver o problema

Como no caso do Problema do Caminho Mínimo, podemos usar um *solver* de Programação Linear Inteira para resolver este problema.

Usando o OpenSolver do Google SpreadSheets para encontrar uma solução para o modelo do exemplo, obtemos a seguinte solução:
$x_{ab} = x_{bd} = x_{dc} = x_{ca} = 1$ e as outras variáveis iguais a 0, o que dá um custo de 13.

Algoritmos para resolver o problema

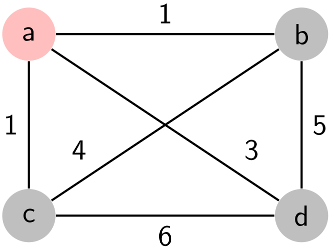
No desenho, a solução obtida é:



Algoritmo heurístico

Podemos também usar uma heurística para resolver este problema. O algoritmo guloso é a opção mais óbvia, que pode dar respostas boas ou muito ruins.

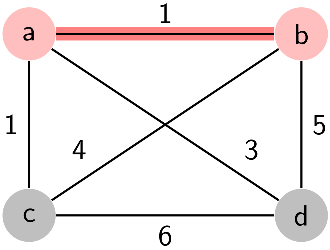
Aplicando o algoritmo guloso no grafo do exemplo passado, a partir do vértice a, temos o seguinte ciclo: $(a,$



Algoritmo heurístico

Podemos também usar uma heurística para resolver este problema. O algoritmo guloso é a opção mais óbvia, que pode dar respostas boas ou muito ruins.

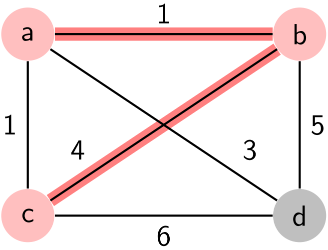
Aplicando o algoritmo guloso no grafo do exemplo passado, a partir do vértice a, temos o seguinte ciclo: $(a, b,$



Algoritmo heurístico

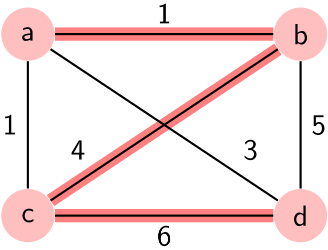
Podemos também usar uma heurística para resolver este problema. O algoritmo guloso é a opção mais óbvia, que pode dar respostas boas ou muito ruins.

Aplicando o algoritmo guloso no grafo do exemplo passado, a partir do vértice a, temos o seguinte ciclo: $(a, b, c,$



Podemos também usar uma heurística para resolver este problema. O algoritmo guloso é a opção mais óbvia, que pode dar respostas boas ou muito ruins.

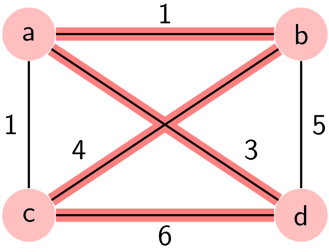
Aplicando o algoritmo guloso no grafo do exemplo passado, a partir do vértice a, temos o seguinte ciclo: $(a, b, c, d,$



Algoritmo heurístico

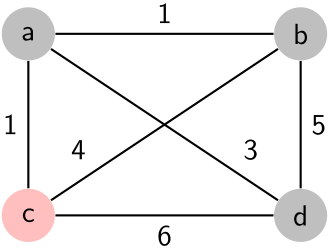
Podemos também usar uma heurística para resolver este problema. O algoritmo guloso é a opção mais óbvia, que pode dar respostas boas ou muito ruins.

Aplicando o algoritmo guloso no grafo do exemplo passado, a partir do vértice a, temos o seguinte ciclo: (a, b, c, d, a), que tem custo 14.



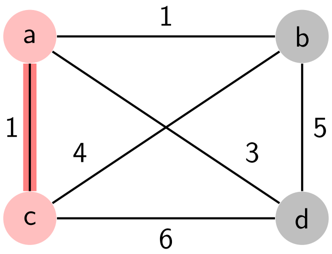
Algoritmo heurístico

Se, para o mesmo exemplo, aplicarmos o algoritmo guloso a partir do vértice c, temos o seguinte ciclo: $(c,$



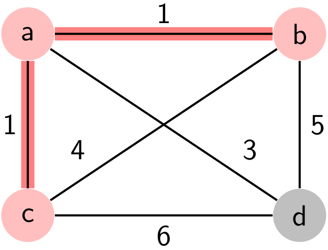
Algoritmo heurístico

Se, para o mesmo exemplo, aplicarmos o algoritmo guloso a partir do vértice c, temos o seguinte ciclo: $(c, a,$



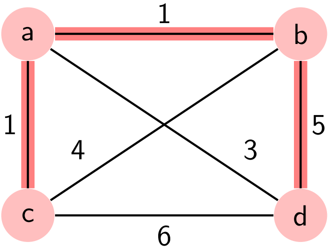
Algoritmo heurístico

Se, para o mesmo exemplo, aplicarmos o algoritmo guloso a partir do vértice c, temos o seguinte ciclo: $(c, a, b,$



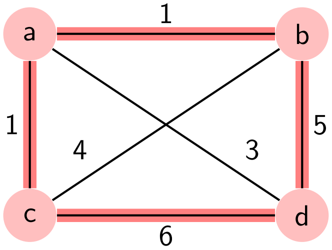
Algoritmo heurístico

Se, para o mesmo exemplo, aplicarmos o algoritmo guloso a partir do vértice c, temos o seguinte ciclo: $(c, a, b, d$



Algoritmo heurístico

Se, para o mesmo exemplo, aplicarmos o algoritmo guloso a partir do vértice c, temos o seguinte ciclo: (c, a, b, d, c), que é ótimo.



Algoritmo exato?

O Problema do Caixeiro Viajante é chamado de NP-difícil. Grosseiramente falando, isso quer dizer que não se conhece nenhum algoritmo polinomial no tamanho do grafo G para determinar um ciclo hamiltoniano de custo mínimo.

Mais do que isso, se for encontrado tal algoritmo, prova-se que $P = NP$, um dos problemas do milênio, que vale um prêmio de 1 milhão de dólares.

Trabalho

Justamente pelo fato do Problema do Caixeiro Viajante ser NP-difícil (além de ter muitas aplicações práticas e teóricas), estuda-se muito diferentes formas de modelá-lo e de resolvê-lo.

O trabalho desta parte da disciplina será, então, o seguinte:

- A sala irá se dividir em grupos de até 4 pessoas.
- Cada grupo deve pesquisar/inventar uma maneira de modelar o Problema do Caixeiro Viajante e uma maneira de resolvê-lo.
- Os modelos e algoritmos devem ser diferentes de um grupo para o outro. E diferentes do que foi apresentado em aula.

Trabalho

- Na próxima aula (dia 19 de março), vocês poderão usar o tempo da aula para fazer o trabalho.
- Na aula seguinte (dia 26 de março), todos os grupos deverão entregar um relatório escrito explicando seu modelo e seu algoritmo, bem como os códigos que foram usados no trabalho.
- Cada grupo terá 15 minutos para apresentar para a turma seu trabalho.
- A nota do trabalho será composta tanto da parte escrita como da apresentação do trabalho.