Roteamento de Veículos

Gustavo Peixoto Silva

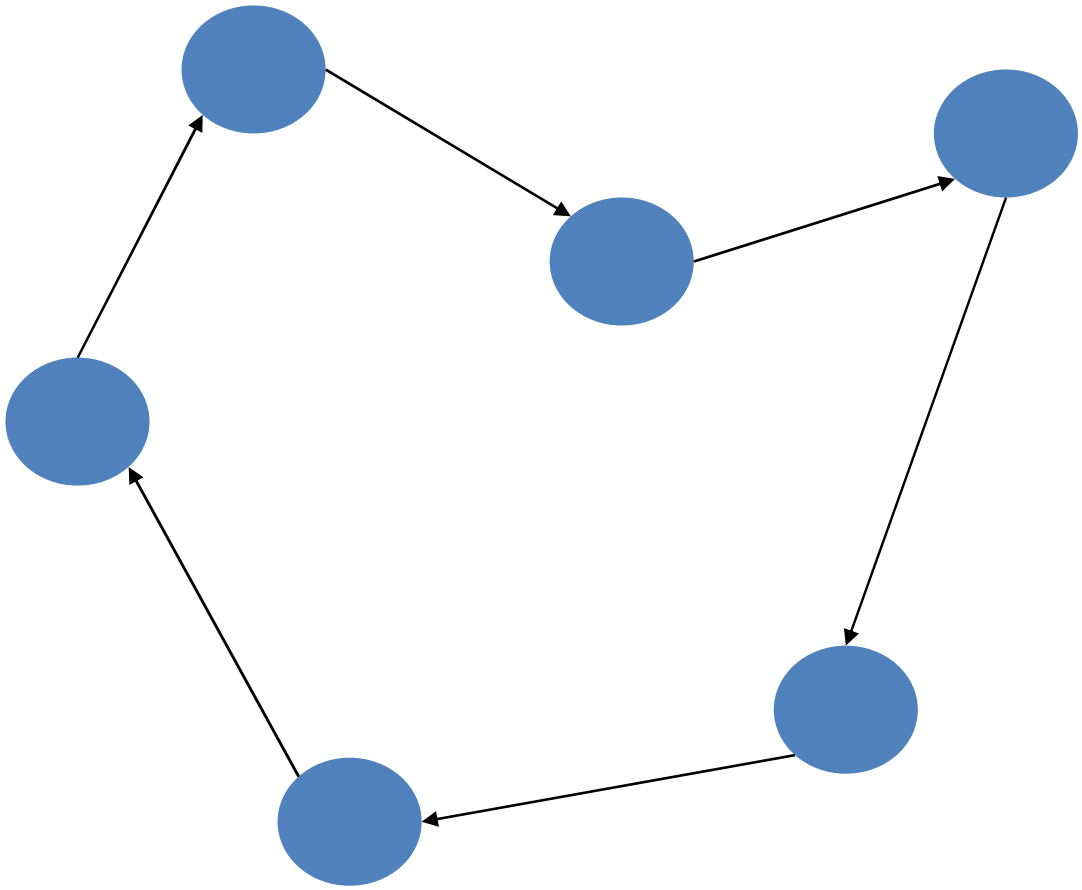
Problema do Caixeiro Viajante

- Dado um conjunto de cidades e uma matriz de distâncias entre elas

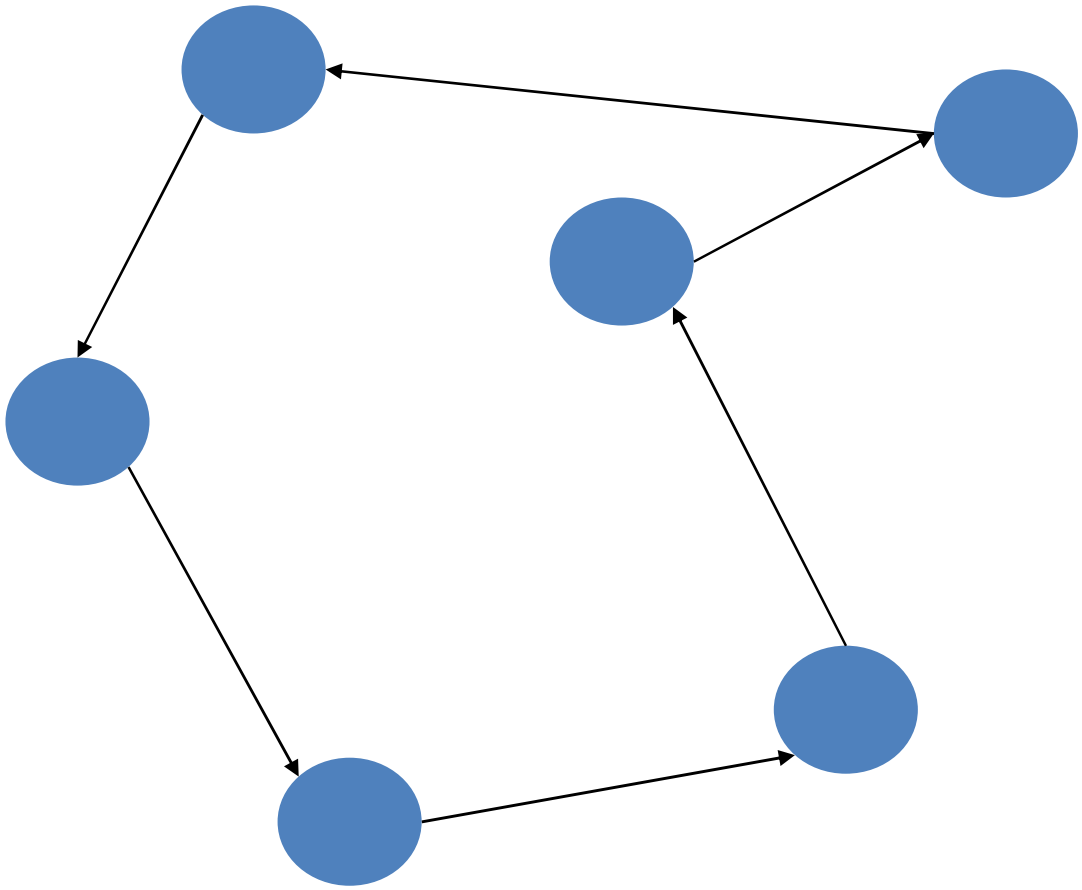
- PCV consiste em encontrar uma rota que:
  - parte de uma cidade origem
  - passe por todas as demais cidades uma única vez
  - retorne à cidade origem ao final do percurso
  - percorra a menor distância possível

- Esta rota é conhecida como Ciclo Hamiltoniano

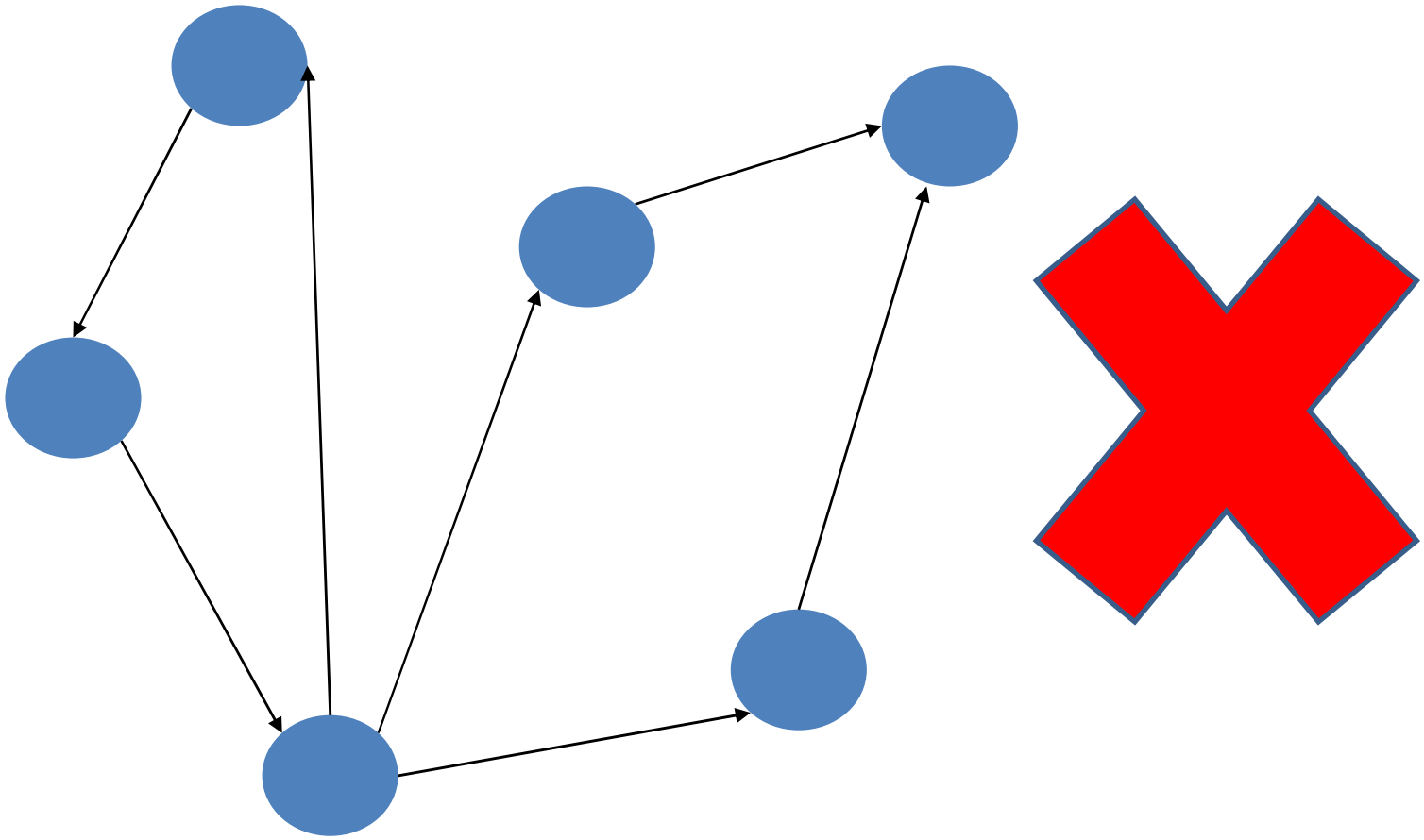
Problema do Caixeiro Viajante



Problema do Caixeiro Viajante

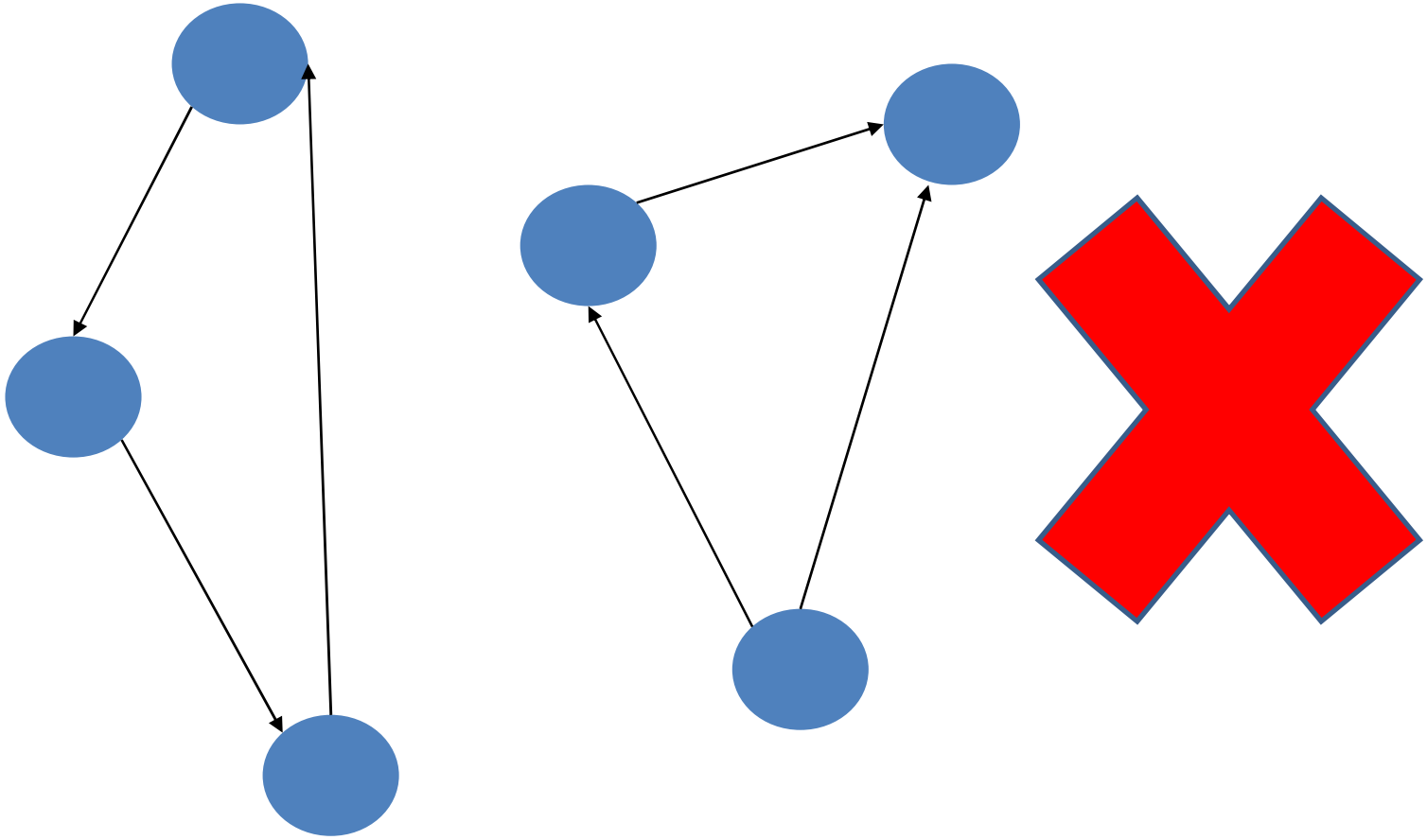


Problema do Caixeiro Viajante



Estas duas rotas não são uma solução para o PCV

Problema do Caixeiro Viajante

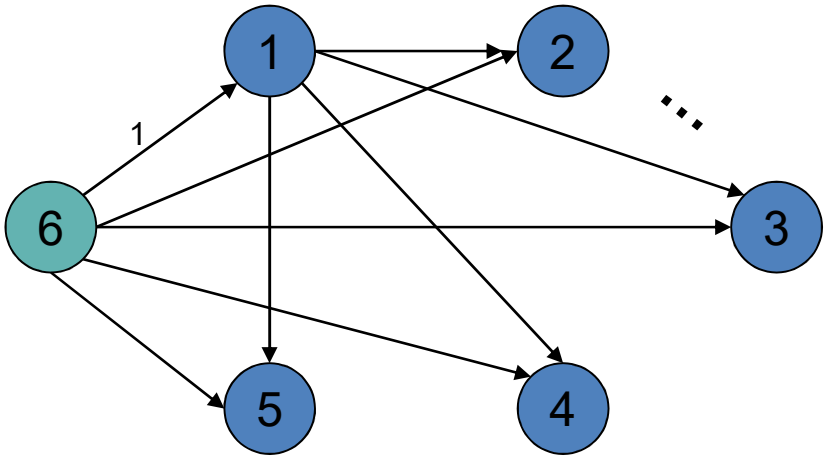


Estas duas rotas não são uma solução para o PCV

Problema do Caixeiro Viajante

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

A Rede que representa o problema é um grafo completo



Formulação Matemática para o Problema do Caixeiro Viajante

- Dados de entrada:
  - Conjunto de cidades
  - d_{ij} = distância entre as cidades i e j
- Variáveis de decisão:
  - X_{ij} = 1 se a aresta (i, j) for usada e 0, caso contrário
  - Y_{ij} = quantidade de fluxo de i para j
- Função objetivo:

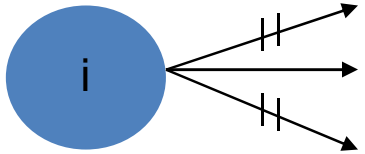
$$\min \sum_{i \in Cidades} \sum_{j \in Cidades} d_{ij} x_{ij}$$

Formulação Matemática para o Problema do Caixeiro Viajante

Restrições:

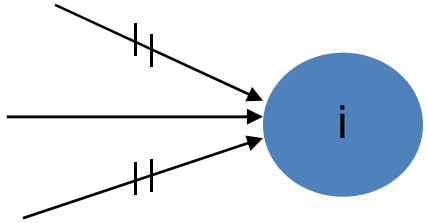
1. De cada cidade i só sai uma aresta:

$$\sum_{j \in Cidades} x_{ij} = 1 \quad \forall i \in Cidades$$



2. A cada cidade j só chega uma aresta:

$$\sum_{i \in Cidades} x_{ij} = 1 \quad \forall j \in Cidades$$



As restrições não garantem que a solução ótima seja sempre uma **rota válida**, assim precisamos adicionar restrições que eliminam sub-rotas, i.e. rotas que são desconectadas.

Existem diferentes maneiras de fazer isto. Vamos ver uma cuja ideia central é a seguinte: Considere que o vendedor carregue consigo os produtos que vende, iniciando no nó 1 com n unidades. Se exigirmos que o vendedor venda exatamente uma unidade em cada ponto, ele precisará percorrer todos os nós para entregar os produtos, assim as subrotas serão eliminadas.

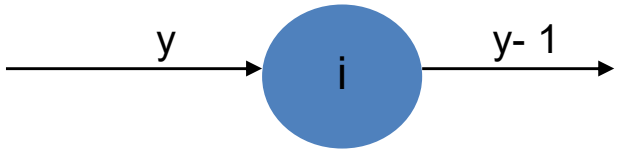
Formulação Matemática para o Problema do Caixeiro Viajante

Seja y_{ij} a quantidade de objetos carregadas do ponto *i* para o ponto *j*

3. O fluxo que chega a uma cidade i menos o que sai é igual a uma unidade:

$$\sum_{j \in Cidades} y_{ji} - \sum_{j \in Cidades} y_{ij} = 1 \quad \forall i \in Cidades \mid i \neq 1$$

$$\sum_{j \in Cidades} y_{ij} - \sum_{j \in Cidades} y_{ji} = n - 1 \quad para \;\; i = 1$$



4. O fluxo máximo em cada aresta é igual a n -1, onde n é o número de cidades:

$$y_{ij} \leq (n-1)x_{ij} \quad \forall i \in Cidades, \forall j \in Cidades$$

5. Integralidade e não negatividade:

$$y_{ij} \geq 0 \quad \forall i \in Cidades, \forall j \in Cidades$$

$$x_{ij} \in \{0,1\} \quad \forall i \in Cidades, \forall j \in Cidades$$

Formulação Matemática para o Problema do Caixeiro Viajante

$$\min \sum_{i \in Cidades} \sum_{j \in Cidades} d_{ij} x_{ij}$$

$$\sum_{j \in Cidades} x_{ij} = 1 \quad \forall i \in Cidades$$

$$\sum_{i \in Cidades} x_{ij} = 1 \quad \forall j \in Cidades$$

$$\sum_{j \in Cidades} y_{ji} - \sum_{j \in Cidades} y_{ij} = 1 \quad \forall i \in Cidades \mid i \neq 1$$

$$\sum_{j \in Cidades} y_{ij} - \sum_{j \in Cidades} y_{ji} = n - 1 \quad para \ i = 1$$

$$y_{ij} \leq (n-1) x_{ij} \quad \forall i \in Cidades, \ \forall j \in Cidades$$

$$x_{ij} \in \{0,1\} \quad \forall i \in Cidades, \ \forall j \in Cidades$$

$$y_{ij} \geq 0 \quad \forall i \in Cidades, \ \forall j \in Cidades$$

```
param n, integer, >= 3;  /* numero de nós */
set V := 1..n;           /* conjunto de nós  */
set E, within V cross V; /* conjunto de arcos */
param c{(i,j) in E};     /* distância do nó i ao nó j */
var x{(i,j) in E}, binary;/* x[i,j] = 1 se o caixeiro viajante for do nó i para o nó  j */
var y{(i,j) in E}, >= 0;  /* y[i,j] é o número de itens que o viajante tem entre o nó i o nó j

minimize total: sum{(i,j) in E} c[i,j] * x[i,j];
/* o objetivo é minimizar a distância total percorrida */

s.t. leave{i in V}: sum{(i,j) in E} x[i,j] = 1;
/* o viajante deixa cada nó uma única vez */

s.t. enter{j in V}: sum{(i,j) in E} x[i,j] = 1;
/* o viajante chega em cada nó uma única vez  */

s.t. node{i in V: i != 1}: sum{(j,i) in E} y[j,i] - sum{(i, j) in E} y[i, j] = 1;
/*  o fluxo que sai de um nó - fluxo que chega no nó =  1 , ou seja, o viajante deixa um item no nó*/

s.t. node1: sum{(1,j) in E} y[1, j] - sum{(j, 1) in E} y[j, 1] = n - 1;
/*  o fluxo que sai da origem - fluxo que chega na origem = n-1. o viajante leva n-1 itens */

s.t. cap{(i,j) in E}: y[i,j] <= (n-1) * x[i,j];
/* se o arco (i,j) não pertence ao caminho mínimo, sua capacidade deve ser igual a zero; c/c. basta ter no
máximo n-1 objetos */
```

Com as restrições de fluxo, y[i,j]

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

Solução ótima 1 -> 2 -> 6 -> 5 -> 4 -> 3 -> 1
Distância = 12

Sem as restrições de fluxo, y[i,j]

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

Cid.	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	6
5	9	7	8	2	0	2
6	1	2	6	6	2	0

Solução ótima
Subrota1: 1 -> 3 -> 1 = custo 2
Subrota2: 2 -> 6 -> 2 = custo 4
Subrota3: 4 -> 5 -> 4 = custo 4
Custo total: 10

Problema do Caixeiro Viajante: Complexidade

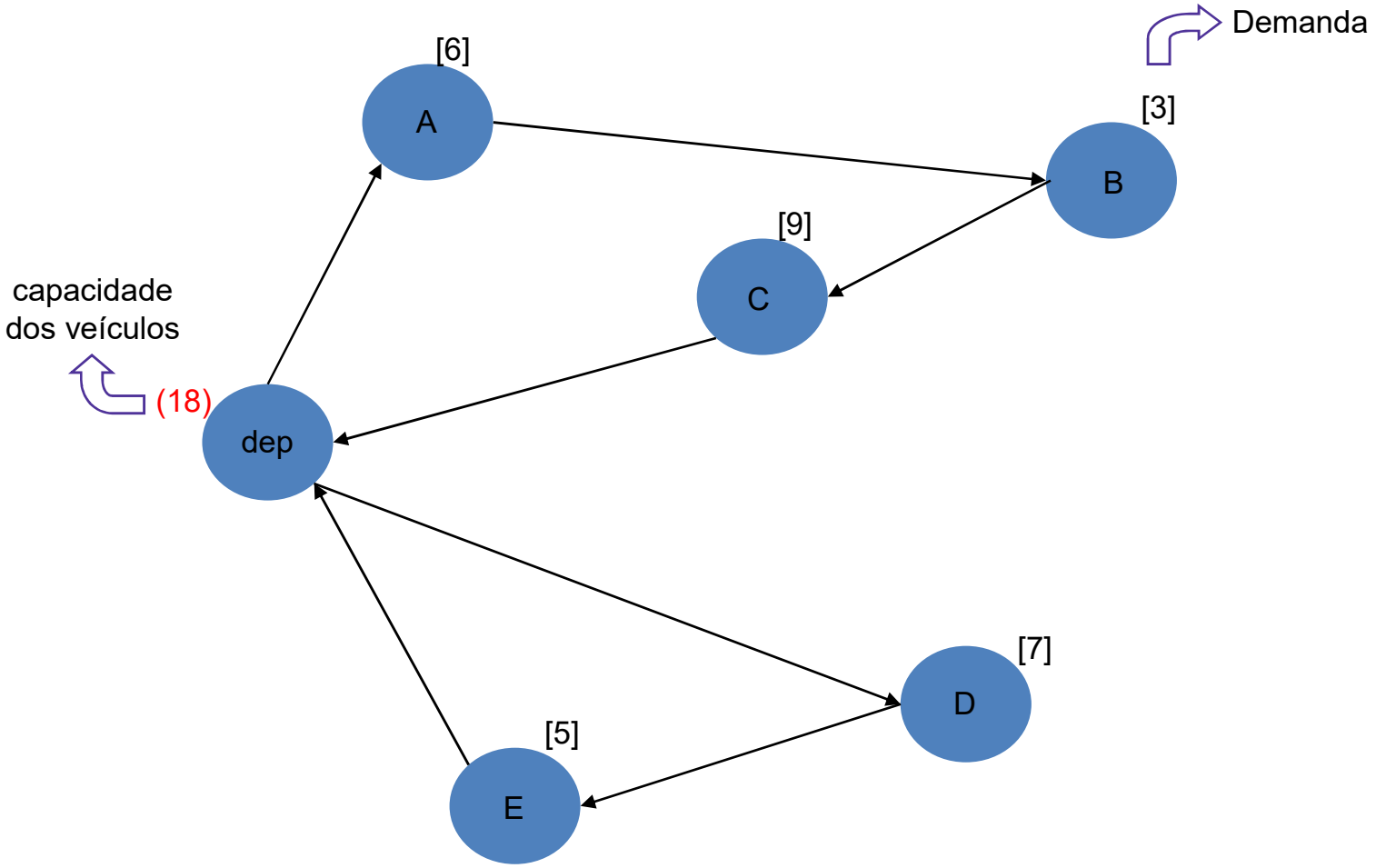
- Considerando PCV simétrico ($d_{ij} = d_{ji}$), para *n* cidades há (*n*-1)!/2 rotas possíveis
- Para *n* = 20 cidades, são 10^{16} rotas possíveis. Um computador que avalia uma rota em 10^{-8} segundos gastaria cerca de 19 anos para encontrar a melhor solução por enumeração completa!
- Métodos de enumeração implícita, tais como *branch-and-bound*, podem reduzir significativamente este tempo
- Mas não há garantia de que isso sempre ocorra

Problema de Roteamento de Veículos

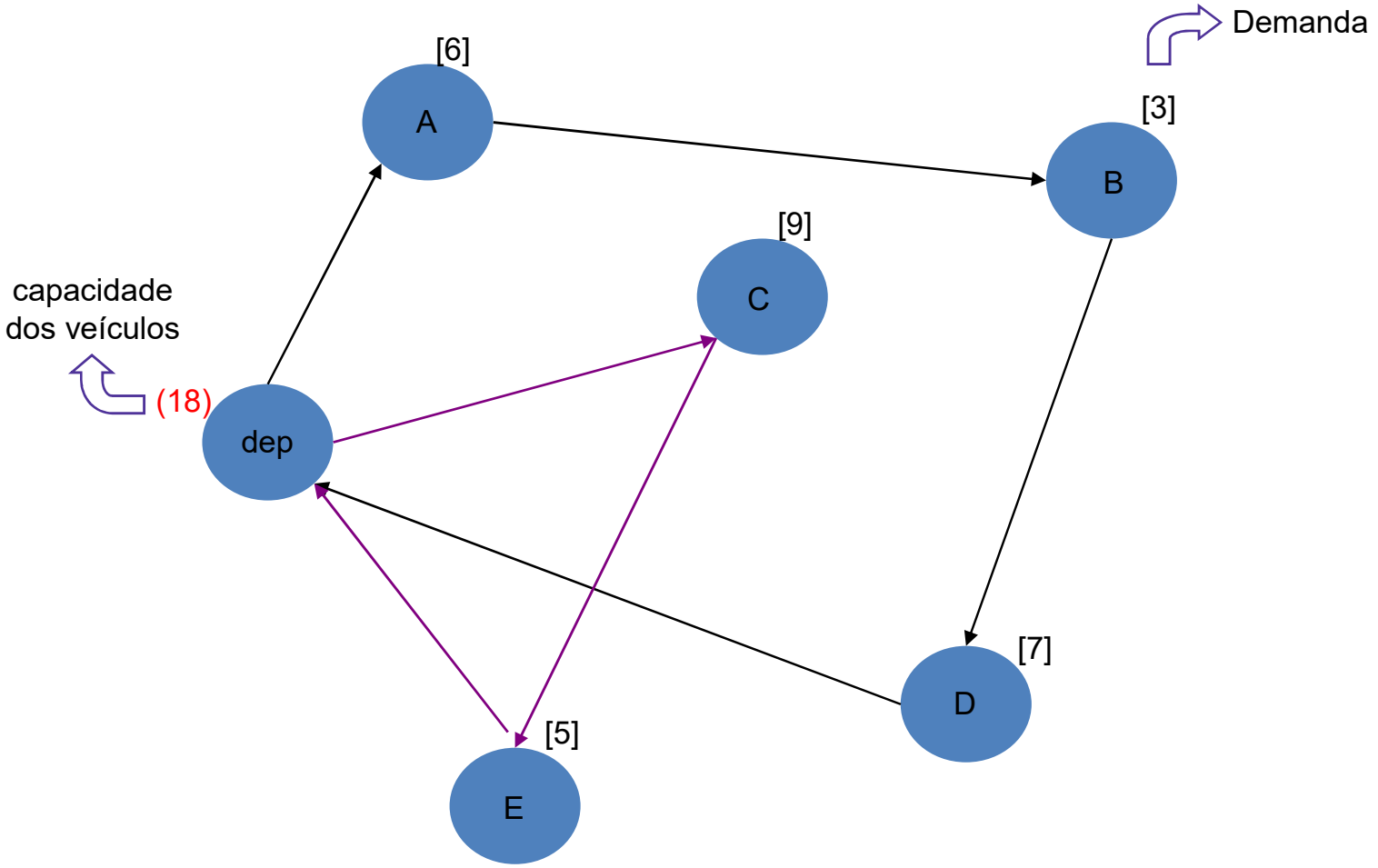
- Dados de entrada:
  - Um depósito
  - Uma frota de veículos, com base no depósito
  - Um conjunto de clientes
  - A demanda dos clientes
  - Uma matriz de distâncias D = (d_{ij}) entre depósito e clientes e entre pares de clientes
  - Cidades = 1 depósito + n clientes

- PRV consiste em encontrar um conjunto de rotas para os veículos tal que:
  - Cada rota comece e termine no depósito
  - Cada cliente seja atendido por um único veículo
  - A capacidade (capVeic) dos veículos seja respeitada
  - A distância total percorrida seja a menor possível
  - O número de veículos seja minimizado

Problema de Roteamento de Veículos



Problema de Roteamento de Veículos



Problema do Caixeiro Viajante - Aplicação

Tinta branca (W), amarela (Y) vermelha (R) e preta (B)

Tempo de limpeza da máquina que as produz, que depende da sequência

	W	Y	R	B
W	0	10	17	15
Y	20	0	19	18
R	50	40	0	25
B	45	40	20	0

tintas → cidades

tempo de limpeza entre 2 tintas → distância.

O problema é determinar o circuito mais curto que passe por todas as cidades (tintas).

Enumeração completa (n-1)!

W Y B R W = 99 W Y R B W = 98

W B Y R W = 124 W B R Y W = 102

W R B Y W = 99 W R Y B W = 124

Para 11 cidades temos 10! = 3.628.800 circuitos!!!

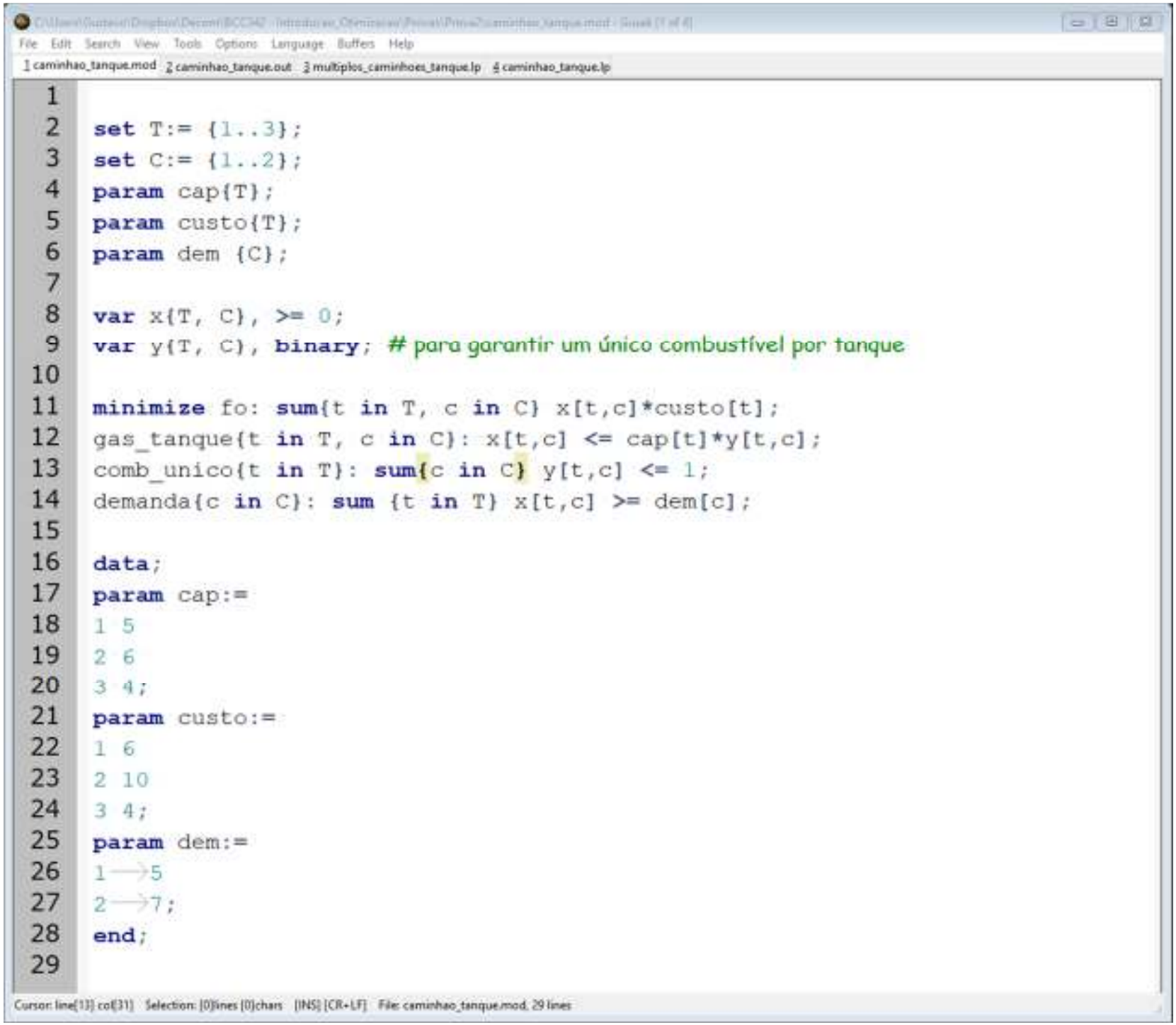
(Brown et al. 1987) Uma empresa usa 4 caminhões–tanques para entregar 4 produtos de gasolina a seus clientes. Cada tanque tem 5 compartimentos com diferentes capacidades: 500, 750, 1.200, 1.500 e 1.750 litros. As demandas diárias dos 4 produtos são estimadas em: 10, 15, 12 e 8 mil litros. Quaisquer quantidades que não possam ser entregues pelos 4 caminhões da empresa devem ser subcontratadas a um custo adicional de 5, 12, 8 e 10 centavos por litro para os produtos 1, 2, 3 e 4, respectivamente. Desenvolver a programação diária ótima de carregamento para os quatro caminhões que minimizará o custo adicional de subcontratação.

(Silva, G. P. 2019) Uma empresa usa **1** caminhão–tanque para entregar **2** produtos de gasolina a seus clientes. Cada caminhão-tanque tem **3** compartimentos com diferentes capacidades: 5, 6 e 4 mil litros. As demandas diárias dos 2 produtos são estimadas em: 5 e 7 mil litros. O custo para cada mil litros de qualquer combustível depende do tanque em que for transportado, sendo de 60$ no tanque 1, 100$ no tanque 2 e de 40$ no tanque 3. Desenvolver a programação diária ótima de carregamento para os quatro caminhões que minimizará o custo adicional de subcontratação.

Minimize fo: + 6 x(1,1) + 6 x(1,2) + 10 x(2,1) + 10 x(2,2) + 4 x(3,1) + 4 x(3,2)
Subject To
gas_tanque(1,1): x(1,1) <= 5 y(1,1)
gas_tanque(1,2): x(1,2) <= 5 y(1,2)
gas_tanque(2,1): x(2,1) <= 6 y(2,1)
gas_tanque(2,2): x(2,2) <= 6 y(2,2)
gas_tanque(3,1): x(3,1) <= 4 y(3,1)
gas_tanque(3,2): x(3,2) <= 4 y(3,2)

comb_unico(1): y(1,1) + y(1,2) <= 1
comb_unico(2): y(2,1) + y(2,2) <= 1
comb_unico(3): y(3,1) + y(3,2) <= 1

demanda(1): x(1,1) + x(2,1) + x(3,1) >= 5
demanda(2): x(1,2) + x(2,2) + x(3,2) >= 7
y(i, j) binário
x(i, j) >= 0


File Edit Search View Tools Options Language Buffers Help
1 caminhao_tanque.mod 2 caminhao_tanque.out 3 multiplos_caminhoes_tanque.lp 4 caminhao_tanque.lp
1
2 set T:= {1..3};
3 set C:= {1..2};
4 param cap{T};
5 param custo{T};
6 param dem {C};
7
8 var x{T, C}, >= 0;
9 var y{T, C}, binary; # para garantir um único combustível por tanque
10
11 minimize fo: sum{t in T, c in C} x[t,c]*custo[t];
12 gas_tanque{t in T, c in C}: x[t,c] <= cap[t]*y[t,c];
13 comb_unico{t in T}: sum{c in C} y[t,c] <= 1;
14 demanda{c in C}: sum {t in T} x[t,c] >= dem[c];
15
16 data;
17 param cap:=
18 1 5
19 2 6
20 3 4;
21 param custo:=
22 1 6
23 2 10
24 3 4;
25 param dem:=
26 1 5
27 2 7;
28 end;
29
Cursor: line[1] col[31] Selection: [0]lines [0]chars [INS] [CR+LF] File: caminhao_tanque.mod, 29 lines