Contents

1 Modelo

Usamos o modelo já conhecido para o TSP convencional como base

$$\begin{aligned} \min & \sum c_e x_e \\ & \sum_{e \in \delta(v)} x_e = 2 v \in V \\ & \sum_{e \in \delta(S)} x_e \leq |S| - 1 S \subset V \end{aligned}$$

No nosso caso, temos que resolver dois TSP's mas que as soluções possuam k arestas em comum.

No nosso modelo x_e^1 indica que usamos a aresta e para o tuor 1 o respectivo para o tuor 2 e D_e indica se a aresta está duplicada.

Nossa função objetivo pode ser a soma dos custos dos dois tuors, ou seja

$$\min \sum_{e \in E} \sum_{i \in \{1,2\}} c_e x_e^i.$$

Repetimos as restrições do TSP para cada um dos tuors.

$$\begin{aligned} & \sum_{e \in \delta(v)} x_e^i = 2 v \in V \ \forall i \in \{1,2\} \\ & \sum_{e \in \delta(S)} x_e^i \leq |S| - 1 \ \forall S \subset V \ \forall i \in \{1,2\} \end{aligned}$$

É importante notar que a segunda equação dá origem a quantidade exponencial de restrições de eliminação de subtuor. No nosso código, podemos

circundar esse problema adicionando as restrições conforme se faz necessário. Assim, quando o modelo termina com um certo conjunto de restrições, podemos conferir, por meio de uma busca de profundidade, se é uma solução viável considerando a restrição de subtuor. Caso não seja, adicionamos as restrições de subtuor que evitam essa solução. Fazemos isso até encontrarmos uma solução viável.

Por fim, adicionamos as restrições que exigem a quantidade de arestas compartilhadas.

$$x_e^i \geq D_e \ \forall e \in E \ \forall i \in \{1,2\}$$
$$\sum_{e \in E} D_e \geq k$$

Assim, nosso modelo final é

$$\min \sum_{e \in E} \sum_{i \in \{1,2\}} c_e x_e^i$$
$$\sum_{e \in \delta(v)} x_e^i = 2 \ \forall v \in V \ \forall i \in \{1,2\}$$
$$\sum_{e \in \delta(S)} x_e^i \leq |S| - 1 \ \forall S \subset V \forall i \in \{1,2\}$$
$$x_e^i \geq D_e \ \forall e \in E \ \forall i \in \{1,2\}$$
$$\sum_{e \in E} D_e \geq k$$

2 Geração de instâncias

Para testar nosso modelo, utilizamos o arquivo de coordenadas disponibilizado pelo professor para calcular nossos custos de arestas. Assim, para instâncias de 100 cidades, utilizamos as 100 primeiras linhas do arquivo.

Durante os testes, modificamos a quantidade de cidades (100, 150, 200 e 250) e o valor de k (zero, metade da quantidade de cidades e a quantidade de cidades).

3 Resultados

Realizamos os testes em um computador equipado de um processador i5 de oitava geração, com 4 cores e 8 threads a 1.6ghz (max boost 3.2) e 8gb de ram, sem swap, com sistema operacional Linux 64bits.

Table 1: Métricas do modelo para as instâncias citadas no formato custo, gap e tempo de execução.

	$k = 0$	$k = \frac{v}{2}$	$k = v$
$v = 100$	(1630, 0%, 12.46)	(2102, 0%, 51.23)	(3463, 0%, 18.29)
$v = 150$	(1966, 0%, 79.01)	(2748, 0%, 565.42)	(4780, 0%, 36.89)
$v = 200$	(2308, 0%, 208.72)	(3458, 0%,)	(6003, 0%, 112.67)
$v = 250$	(2916, 11.1%, 1113)	(7525, 0%,)	(6999, 0%,)

É importante ressaltar que a instância com $v = 250$ e $k = 0$ foi finalizada pelo sistema operacional por falta de memória ram. O resultado reportador aqui é da última atualização fornecida pelo Gurobi.

4 Análise

Observando as métricas obtidas, vemos os piores tempos de execução são encontrados quando $k = \frac{v}{2}$.

5 Fontes

`https://colab.research.google.com/github/Gurobi/modeling-examples/blob/master/traveling_salesman/tsp_gcl.ipynb`