Submetido para MO824

Atividade 2 - Grupo 8 - MO824

Victor Ferreira Ferrari - RA 187890
Flávio Murilo Reginato - RA 197088
Vitor Satoru Machi Matsumine - RA 264962

Resumo. Este documento propõe uma solução para o problema 2-TSP, apresentando uma modelagem matemática para o problema linear, realizando ensaios através do software Gurobi. Essa solução é então avaliada em comparação com a solução do problema TSP. Todas as instâncias testadas foram resolvidas de maneira ótima, utilizando *lazy constraints*. O custo da solução do TSP multiplicado por 2 é um limitante inferior para o custo da solução do 2-TSP para a mesma instância, porém a qualidade desse limitante pode diminuir a utilidade (*gap* de em média 19%).

Palavras-chave. Programação Linear, Otimização, Gurobi, 2-TSP.

1. Introdução

Este trabalho consiste na apresentação de um modelo matemático e na realização de experimentações para o problema proposto na Atividade 2 de MO824 2S-2020. Chamado 2-TSP, o problema é uma variação do tradicional TSP (*travelling salesman problem*) e busca encontrar dois ciclos hamiltonianos com conjunto de arestas disjuntos e cuja soma de peso dessas arestas seja mínima.

Para isso, foi empregada programação linear utilizando *lazy constraints* através do software Gurobi, onde o problema foi modelado baseando-se na implementação que a própria documentação do software fornece para o TSP.

2. Modelo Matemático

As seguintes variáveis foram usadas no modelo:

x_e: Variável de decisão binária associada à presença ($x_e = 1$) ou não ($x_e = 0$) da aresta e no primeiro ciclo.

y_e: Variável de decisão binária associada à presença ($y_e = 1$) ou não ($y_e = 0$) da aresta e no segundo ciclo.

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e + c_e y_e & & (1)\\
\text{s.a} \quad & \sum_{e \in \delta(i)} x_e = 2 & \forall i \in V \quad & (2)\\
& \sum_{e \in \delta(i)} y_e = 2 & \forall i \in V \quad & (3)\\
& \sum_{e \in E(S)} x_e \leq |S| - 1 & \forall S \subset V \quad & (4)\\
& \sum_{e \in E(S)} y_e \leq |S| - 1 & \forall S \subset V \quad & (5)\\
& x_e + y_e \leq 1 & \forall e \in E \quad & (6)\\
& x_e, y_e \in \{0, 1\} & \forall e \in E \quad & (7)
\end{aligned}
$$

Onde $\delta(i)$ é o conjunto de arestas incidentes no vértice i, $S \subset V$ é um subconjunto próprio de vértices e $E(S)$ é o conjunto das arestas cujas extremidades estão em S.

O modelo é uma generalização do programa linear inteiro correspondente ao TSP, mas difere-se dele pelo objetivo, que busca minimizar o custo de dois ciclos hamiltonianos disjuntos (equação 1), e também pelas seguintes restrições:

Restrições (2-3): garante que todos os vértices dos ciclos tenham apenas duas arestas incidentes.

Restrições (4-5): elimina subciclos ilegais nos ciclos. Podem ser implementadas por *lazy constraints*, já que o número de restrições é exponencial.

Restrição (6): assegura que os conjuntos de arestas dos dois ciclos serão disjuntos.

Esse modelo introduz um novo conjunto de variáveis para o segundo ciclo. Porém, é possível reduzir para um único conjunto de variáveis, com uma dimensão extra. Desse modo, o modelo é facilmente escalável para encontrar k ciclos (*k-TSP*). O modelo foi feito desta maneira por legibilidade.

3. Metodologia

3.1. Gerador de Instâncias

As instâncias foram geradas de maneira aleatória e uniforme a partir de uma quantidade de cidades fornecida e uma *random seed*. Como proposto, as cidades foram posicionadas aleatoriamente no plano, e as distâncias euclideanas entre os pares foram calculadas. A geração é feita no mesmo arquivo que a execução, e todas as instâncias foram executadas com a mesma *seed*.

3.2. Especificações do Computador

As especificações de hardware do computador no qual foram feitas as execuções estão na Tabela 1.

Tabela 1: Condições de Execução

Modelo da CPU	Intel(R) Core(TM) i7-9750H (6C/12T)
Frequência do Clock da CPU	2.60 GHz
RAM	16 GB/2660 MHz

O sistema operacional utilizado foi o Windows 10 (64 bits). Foi utilizado como software de execução o *solver* Gurobi Optimizer V9.0.3 empregando 12 threads. Os modelos foram executados com limite de 1800 segundos (30 minutos) e sem limite de memória.

4. Resultados Obtidos e Análise

Para cada tamanho proposto na atividade foi gerada uma instância e os resultados alcançados estão na Tabela 2.

Tabela 2: Resultados Obtidos

Número de cidades	Quantidade de variáveis TSP	Custo da solução TSP (multiplicado por 2)	Tempo de Execução TSP [s]	Quantidade de variáveis 2-TSP	Custo da solução 2-TSP	Tempo de Execução 2-TSP [s]
20	190	8.26110	0.03290	380	10.4170	0.03989
40	780	11.81582	0.16655	1560	14.0863	0.54155
60	1770	13.21266	0.16655	3540	16.0766	1.89393
80	3160	14.90658	0.69813	6320	18.5450	14.00453
100	4950	16.07044	1.07711	9900	20.2082	20.55900

Pela Tabela 2, temos uma comprovação do comportamento esperado. O custo de uma solução do 2-TSP é no mínimo o dobro do custo da solução do TSP para a mesma instância, no caso em que ambos ciclos têm o mesmo custo, e esse custo é o mínimo. Na maioria dos casos, assim como em todas as instâncias testadas, isso não ocorreu, então o custo do 2-TSP foi maior.

Além disso, os tempos de execução das instâncias do TSP foram menores que os do 2-TSP. Isso era esperado, porém o aumento dos tempos no 2-TSP seguiu uma curva consideravelmente mais inclinada que no problema simples, indicando assim que o problema tem uma dificuldade maior.

Pelas relações entre os problemas discutidas nesta seção, pode-se concluir que o custo do TSP multiplicado por 2 pode ser utilizado como **limitante inferior** para o 2-TSP. Para instâncias maiores, o tempo necessário para o problema simples é suficientemente pequeno para ser possivelmente vantajoso como limitante inferior inicial. A principal desvantagem é na qualidade desse limitante, que pode variar. O *gap* entre as soluções foi de em média 19%, suficientemente grande para diminuir a utilidade.