Atividade 2 - Programação Linear com Lazy Constraints

Felipe Pereira - RA: 263808
José Antonio Mauad Leis - RA: 219061
Lucas Guesser Targino da Silva - RA: 203534

16 de abril de 2022

1 Enunciado do Problema

Sejam:

1. $G = \langle V, E \rangle$: um grafo não-orientado completo:
   (a) V: conjunto de vértices;
   (b) E: conjunto das arestas;
2. $c^1 c^2 : E \to \mathbb{R}_+$ duas funções custo nos vértices;
   (a) dada uma aresta e, escrevemos $c^1(e) = c^1_e$ e $c^2(e) = c^2_e$;
3. k: parâmetro de similaridade de ciclos;

Objetivo: encontrar dois ciclos Hamiltonianos com custo total mínimo, tal que pelo menos k arestas do grafo sejam visitadas por ambos os ciclos.

2 Modelo Matemático

2.1 Variáveis de Decisão

- X^1_e: presença da aresta e no primeiro ciclo;
- X^2_e: presença da aresta e no segundo ciclo;
- D_e: presença de duplicação da aresta e;

Todas as variáveis de "presença" são decisões binárias com a seguinte interpretação de valores:

0 : ausente

1 : presente

2.2 Problema de Otimização

Minimizar:

$$\sum_{e \in E} c_e^1 \; X_e^1 + \sum_{e \in E} c_e^2 \; X_e^2 \tag{1}$$

Sujeito a:

$$\sum_{e \in \delta(v)} X_e^1 = 2 \qquad \forall v \in V \tag{2}$$

$$\sum_{e \in \delta(v)} X_e^1 = 2 \qquad \forall v \in V \tag{3}$$

$$\sum_{e \in E(S)} X_e^1 \leq |S| - 1 \qquad \forall S \subseteq V, S \neq V, S \neq \emptyset \tag{4}$$

$$\sum_{e \in E(S)} X_e^2 \leq |S| - 1 \qquad \forall S \subseteq V, S \neq V, S \neq \emptyset \tag{5}$$

$$X_e^1 + X_e^2 \leq 2 \; D_e \qquad \forall e \in E \tag{6}$$

$$\sum_{e \in E} D_e \geq k \tag{7}$$

$$X_e^1, X_e^2, D_e \in \{0, 1\} \qquad \forall e \in E \tag{8}$$

2.3 Explicação das Restrições

- A função objetivo (1) é soma do custo de todas as arestas selecionadas.
- As restrições (2) e (3) garantem que a quantidade de arestas incidentes em todos os vértices seja 2, nos ciclos 1 e 2 respectivamente. Essa condição faz com que todos os vértices tenham que ser visitados (duas arestas pois uma é a de "entrada" e a outra a de "saída").
- As restrições (4) e (5) garantem que não existam subciclos nos ciclos. Nessas restrições, S é um subconjunto próprio e não-vazio dos vértices do problema. A expressão $E(S)$ é o conjunto das arestas cujos vértices (ambos) estão em S.
- A restrição (6) garante que, se uma aresta foi escolhida para ser duplicada, então essa aresta aparecerá nos dois ciclos.
- A restrição (8) garante que todas as variáveis são decisões binárias, ou seja, assumem apenas um de dois possíveis valores: 0 e 1.

2.4 Tamanho das Restrições

- Restrições (2) e (3): uma para cada vértice. Total: $2 \cdot |V|$;
- Restrições (4) e (5): uma para $S \in \mathcal{P}(V), S \neq V, S \neq \emptyset$. Total: $2 \cdot \left(2^{|V|} - 2\right)$;
- Restrições (6): uma para cada aresta. Total: $|E| = \dfrac{|V|^2 - |V|}{2}$ (já que o grafo é completo);
- Restrições 7: apenas uma. Total: 1;

Assim, o número total de restrições é:

$$T_r = 2 \cdot |V| + 2 \cdot \left(2^{|V|} - 2\right) + \frac{|V|^2 - |V|}{2} + 1 \tag{9}$$

Note que $T_r \in \mathcal{O}\left(2^{|V|}\right)$, isto é, há um número exponencial de restrições.

Na implementação computacional do problema, não é possível adicionar tantas restrições por falta de recursos computacionais (memória e processamento). Há entretanto uma forma de contornar o problema atŕavés do que se chama de *lazy evaluation*. A ideia é não adicionar tais restrições no início. Conforme soluções factíveis são encontradas, verifica-se se há subciclos nelas e, caso sim, adiciona-se apenas as restrições necessárias para eliminar tais subciclos.

Dependendo do caso em mãos, essa abordagem pode reduzir drasticamente o número de restrições e consequentemente acelerar a busca.

3 Experimento Computacional

3.1 Configuração da Máquina

O problema foi executado num ideapad S145 81S90005BR: Lenovo IdeaPad S145 Notebook Intel Core i5-8265U (6MB Cache, 1.6GHz, 8 cores), 8GB DDR4-SDRAM, 460 GB SSD, Intel UHD Graphics 620.

O sistema operacional foi o Fedora 35 executando o Python 3.7.12 e Gurobi Optimizer v9.5.1rc2[1].

Como linguagem de programação, utilizamos Python[2] pela facilidade de uso e disponibilidade de ferramentas.

3.2 Dados do Problema

Os dados do problema foram fornecidos em um arquivo contendo 4 colunas e 250 linhas. A interpretação dos dados é a seguinte: cada linha representa um vértice e cada par de coluna as coordenadas desse vértice. A razão para um vértice ter duas posições diferentes é simplesmente para que as distâncias entre eles tenham valores diferentes no primeiro e no segundo ciclo.

O modelo na verdade precisa apenas de pesos. Construímos a primeira função de custo como a distância euclidiana entre os pontos das colunas 1 e 2. Da mesma forma, utilizamos distância euclidiana entre os pontos das colunas 3 e 4 para construir a segunda função de custo.

Note que, com essa interpretação, parece que temos dois conjuntos de vértices, um definido pelas colunas 1 e 2, e outro definido pelas colunas 3 e 4. Conforme explicitado no primeiro parágrafo da seção, esse não é o caso. Cada linha é um vértice e os valores fornecidos servem apenas para calcular a distância euclidiana e usá-la como peso para as arestas. Dessa forma, a aresta que liga os vértices representados pelas linhas 12 e 84, por exemplo, possui dois pesos diferentes, um para ser utilizado no primeiro ciclo e outro para ser utilizado no segundo.

3.3 Geração das Instâncias

Para gerar instâncias de um dado tamanho N, utilizamos as primeiras N linhas dos dados fornecidos.

Quantidade de Vértices	Similaridade	Soluções Testadas	Soluções Factíveis	Custo da Solução	Gap de Otimalidade	Tempo de Execução (s)	Limite de Tempo (s)
100	0	12	7	1536,97	4,08	3,55	1800
100	50	12	10	2001,67	0	60,71	1800
100	100	12	3	3353,19	0	52,29	1800
150	0	12	3	1829,19	0	30,88	1800
150	75	12	5	2595,37	0	678,63	1800
150	150	12	1	4612,43	0	15,51	1800
200	0	12	1	2127,86	0	64,82	1800
200	100	12	1	14310,67	0,77	1800	1800
200	200	12	4	5785,59	0	242	1800
250	0	12	10	2379,07	7,43	512,57	1800
250	125	12	0	N/A	N/A	1800	1800
250	250	12	0	N/A	N/A	1800	1800

Figura 1: Resultados da Execução

	Custo Mínimo	
# Vértices	K	Custo
100	0	1536,97
150	0	1829,19
200	0	2127,86
250	0	2379,07
100	0.5	2001,67
150	0.5	2595,37
200	0.5	14310,67
250	0.5	N/A
100	1	3353,19
150	1	4612,43
200	1	5785,59
250	1	N/A

Tabela 1: Custo da soluções por Vértice/Similaridade

4 Resultados

4.1 Resultados Obtidos

O programa foi construído para processar todas as instâncias e parâmetros pedidos no exercício. Mas devido a restrições computacionais, não conseguimos executar os dois últimos (Vertíces-250 & Similaridade-0.5, Vértices-250 & Similaridade-1). Abaixo podemos ver a Tabela 1 dos resultados obtidos.

Podemos notar que, mesmo com o k similar, o aumento no número de Vértices também gera aumento do custo da solução, o que já era esperado pela teoria, conforme demonstrado na Tabela 1:

Com isso, foi possível compilar o custo da solução, de acordo com os parâmetros de Vértices e Similaridade, conforme a Figura 2

Assim como a relação do aumento de complexidade com o Tempo de Execução, conforme a Figura 3

5 Análise e Observações Finais

Podemos notar, ao avaliar o resultado das observações, que, para cada conjunto de vértices (em nosso caso: 100, 150, 200 e 250) temos um custo crescente com a adição de complexidade, ou seja, tanto com a adição de vértices, quanto com o aumento da similaridade (onde k pode ser

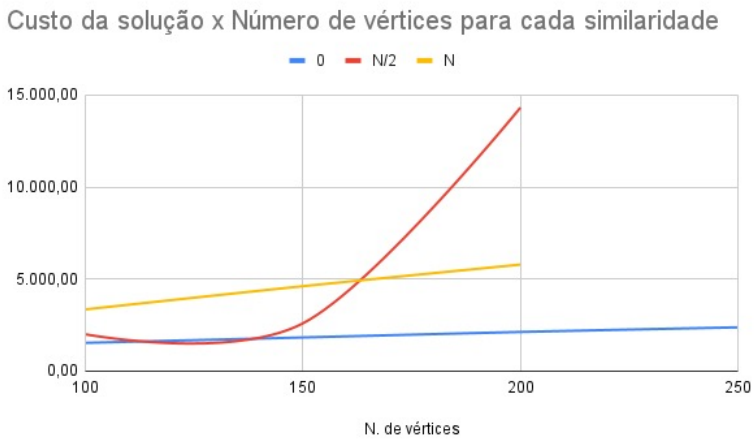

Figura 2: Resultados da Execução

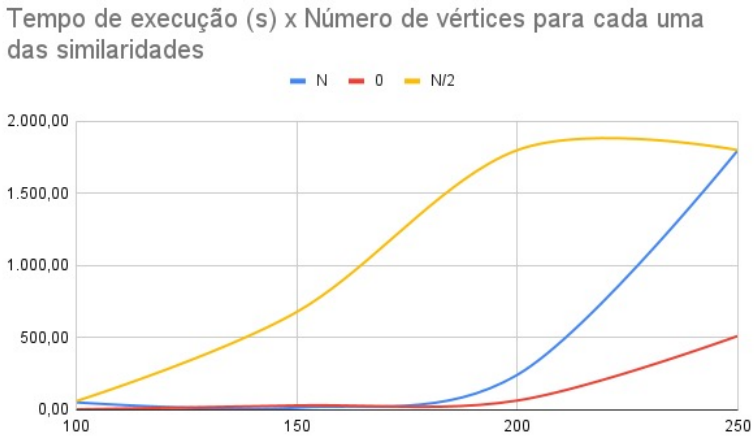

Figura 3: Resultados da Execução

igual a 0, 0.5, 1).

O aumento do custo da solução parece guardar uma relação linear com o aumento de vértices, embora fosse possível especular que a relação fosse exponencial a priori, dado o aumento da complexidade da solução. O mesmo é válido para o aumento da similaridade, representada pela variável k. O aumento do parâmetro de similaridade parece guardar uma relação linear de aumento com o custo da solução obtida.

O mesmo não pode ser dito em relação ao tempo computacional gasto para calcular as soluções. O tempo cresce exponencialmente de acordo com o aumento da complexidade da soluções, que é dada pela quantidade de vértices e pela similaridade entre os dois ciclos.

Este tipo de comportamento fez com que não conseguíssemos calcular as soluções para k=0.5 e k=1 para V=250. É importante notar que o enunciado do exercício nos colocou uma restrição de tempo de execução da busca por soluções em 30 minutos, ou seja, dentro deste tempo, o algoritmo não foi capaz de encontrar soluções viáveis para o problema. Testes com tempos maiores são recomendados para que se encontrem os resultados destas combinações.

Referências

[1] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2022.

[2] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.