Submetido para disciplina MO824

Atividade 3 - Relaxação Lagrangiana

Felipe Domingues - RA 171036

Sandro Henrique Uliana Catabriga - RA 219597

1. Introdução

1.1. Descrição do problema

O objetivo desta atividade consiste na aplicação da técnica de relaxação Lagrangiana, resolvida pelo método dos subgradientes, para a obtenção de limitantes primais e duais de um problema de programação linear inteira. Para tal, iremos considerar o problema dos caixeiros viajantes k-semelhantes.

1.2. kSTSP

Seja um grafo não-orientado completo $G(V, E)$, onde V é o conjunto dos vértices e E é o conjunto das arestas. Em cada aresta $e \in E$ há dois custos c_e^1, c_e^2: $E\vec{R}+$. O objetivo do problema consiste em encontrar dois ciclos Hamiltonianos com custo total mínimo, tal que pelo menos k arestas do grafo sejam visitadas por ambos os ciclos. O parâmetro k define a similaridade entre os ciclos, de modo que $k = |V|$ implica que uma solução factível corresponda a dois ciclos Hamiltonianos contendo as mesmas $|V|$ arestas, enquanto $k = 0$ resulta em uma solução factível ser qualquer par de ciclos Hamiltonianos.

2. Modelo Matemático

Sejam as seguintes variáveis para o kSTSP:

- x_e^k é uma variável de decisão binária associada à presença ($x_e^k = 1$) ou não ($x_e^1 = 0$) da aresta e na rota do caixeiro k;

- z_e é uma variável de decisão binária associada a presença ($z_e = 1$) ou não ($z_e = 0$) da aresta e nas rotas de ambos caixeiros.

Considera também as seguintes constantes e informações:

- c_e^k refere-se ao custo da aresta e na rota do caixeiro k;
- $\delta(i)$ é o conjunto de arestas que incidem no vértice i;
- $S \in V$ é um subconjunto próprio de vértices;
- $E(S)$ é o conjunto das arestas cujos dois vértices terminais estão em S.
- k é uma variável inteira que define a similaridade entre os dois ciclos, ou seja, k é o número de arestas que são visitadas por ambos os caixeiros.

A partir disso, podemos considerar o seguinte modelo linear para solucionar o problema:

$$\text{(P)} \quad \text{minimize} \sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k \cdot x_e^k \tag{2.1}$$

$$\text{sujeito a} \quad \sum_{e\in\delta(i)} x_e^k = 2 \qquad \forall\, i \in V, \forall k \in \{1,2\} \tag{2.2}$$

$$\sum_{e\in E(S)} x_e^k \leq |S| - 1 \qquad \forall\, S \subset V, \forall k \in \{1,2\} \tag{2.3}$$

$$x_e^k \geq z \qquad \forall\, e \in E, \forall k \in \{1,2\} \tag{2.4}$$

$$\sum_{e\in E} z_e \geq k \tag{2.5}$$

$$x_e^k, z_e \in \{0,1\} \qquad \forall\, e \in E, \forall k \in \{1,2\} \tag{2.6}$$

A função objetivo (2.1) minimiza o custo da solução, composto pela soma dos custos de todas as arestas utilizadas nos ciclos. As restrições são as seguintes:

1. O conjunto de restrições (2.2) diz que cada vértice deve possuir duas arestas incidentes, para o ciclo de cada um dos dois caixeiros;
2. O conjunto de restrições (2.3) visa a eliminação de subciclos ilegais, ou seja, a rota de cada caixeiro deve corresponder a um único ciclo que visita todos os vértices;

3. O conjunto de restrições (2.4) garante que, caso $z_e = 1$, então a aresta e estará presente nos ciclos dos dois caixeiros;

4. A restrição (2.5) força pelo menos k arestas em comum nos ciclos dos dois caixeiros;

5. Por fim, (2.6) diz respeito ao domínio das variáveis.

3. Modelo Matemático de Relaxação Lagrangiana

Baseando-se no modelo linear apresentado anteriormente, para produzir o modelo de relaxação Lagrangiana decidimos relaxar a restrição (2.4). Dessa forma, obtemos o seguinte problema Lagrangiano para limitantes inferiores:

porque negativo?

$$\text{(LLBP)} \quad \text{minimize} \sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k \cdot x_e^k - \sum_{k\in\{1,2\}} \sum_{e\in E} \lambda_e^k (z_e - x_e^k) \tag{3.1}$$

Sendo os multiplicadores de Lagrange λ vetores de valores não-negativos.

O modelo relaxado Lagrangeano está sujeito ainda às seguintes restrições: (2.2), (2.3), (2.5) e (2.6).

Aplicando as devidas transformações algébricas, obtemos a seguinte função objetivo para o problema relaxado Lagrangeano:

$$\text{(LLBP)} \quad \text{minimize} \sum_{k\in\{1,2\}} \sum_{e\in E} (c_e^k - \lambda_e^k) x_e^k + \sum_{e\in E} (\lambda_e^1 + \lambda_e^2) z_e \tag{3.2}$$

Considerando as variáveis de decisão envolvidas, as restrições (2.2), (2.3) se aplicam à parte em vermelho, e a restrição (2.5) à parte em azul. Dessa forma podemos decompor em três problemas independentes, sendo a parte em vermelho dois TSP's, e a parte em azul um problema de minimizar os valores de z, tendo como fator na função objetivo a soma dos valores λ^1 e λ^2 (resolução por inspeção).

É importante notar que, quando $k = 0$, a restrição (2.5) pode ser ignorada, e pode-se considerar $z_e = 0$, $\forall e \in E$. Dessa forma, resolver os dois TSP's independentemente, com $\lambda_e^k = 0$, traz a solução ótima para o problema original.

3.1. Dual Lagrangiano

Considerando que o problema (LLBP), para valores não-negativos de λ, nos permite encontrar um limitante inferior para o problema, o problema Dual Lagrangiano (DL) objetiva encontrar os melhores valores para λ, de forma a obter resultados cada vez mais próximos da solução ótima.

estão faltando as restrições do problema

$$\text{(DL)} \quad \text{maximize } \{ \text{ minimize } \sum_{k\in\{1,2\}} \sum_{e\in E} (c_e^k - \lambda_e^k) x_e^k + \sum_{e\in E} (\lambda_e^1 + \lambda_e^2) z_e \} \tag{3.3}$$

$$\text{sujeito a} \quad \lambda_e^k \geq 0 \qquad \forall\, e \in E, \forall k \in \{1,2\} \tag{3.4}$$

3.2. Método do Subgradiente

Para resolver o Dual Lagrangiano (DL), obtendo limitantes inferiores cada vez mais altos, utilizamos o método do subgradiente. O pseudoalgoritmo para o método do subgradiente pode ser definido por:

1. Escolha valores iniciais não-negativos para os multiplicadores λ para a iteração $k = 1$
2. Resolver TSP^1, TSP^2 e o problema por inspeção com os valores atuais de λ, somar as três soluções obtendo a solução x^k
3. Utilizar uma heurística Lagrangiana para obter o limitante superior válido Z_{UB} a partir da solução calculada
4. Calcular o subgradiente g_i^k para cada restrição relaxada
5. Calcular um valor α para o passo
6. Atualizar λ utilizando o passo α calculado anteriormente

O subgradiente está relacionado com a penalidade gerada na função objetivo pela restrição relaxada. Nesse caso, na etapa 4 o gradiente g_e^k é definido por:

$$g_e^k = z_e - x_e^k \qquad \forall k \in \{1,2\}, \forall e \in E$$

considerando um fator positivo π, o passo α para o nosso modelo pode ser calculado, em cada iteração k:

$$\alpha^{(k)} = \pi \cdot ((Z_{UB} - Z_{LB}^k)/(\sum_{e \in E}(g_e^1)^2 + (g_e^2)^2)$$

Sendo Z_{UB} o melhor limitante superior encontrado com base na Heurística Lagrangiana

Atualização do λ?

3.3. Heurística Lagrangiana

A Heurística Lagrangeana tem como objetivo tomar um limitante inferior calculado durante as iterações no método do subgradiente e produzir um limitante superior válido para o problema original. No nosso exemplo, dada uma solução com x_e^1 e x_e^2 sendo as rotas dos caixeiros, e os custos c_e^1 e c_e^2, a seguinte heurística foi implementada:

$$Z_{UB} = min(\sum_{e \in E}(c_e^1 + c_e^2) \cdot x_e^1, \sum_{e \in E}(c_e^1 + c_e^2) \cdot x_e^2)$$

De forma resumida, obtivemos um limitante superior válido ao considerar a mesma rota (x_e^1 ou x_e^2) para ambos caixeiros, tomando aquela que tiver menor custo quando aplicado para ambos.

→ Problema dessa heurística é que ela desperdiça completamente a informação de uma das rotas.

4. Materiais e métodos

4.1. Configurações e Código

Para os experimentos foi utilizada uma máquina com chip Apple M1 e 8GB de memória RAM. A implementação do modelo linear, do modelo linear relaxado e do TSP foi feita utilizando a API em C++ do software Gurobi na versão 9.5.1, disponibilizando os 8 núcleos físicos da máquina para execução. Não foi utilizado nenhum parâmetro especial para execução dos programas. O modelo de relaxação Lagrangiana e a heurística Lagrangiana também foram implementados na linguagem C++, utilizando compilador $g++$ 13.1.6.

A solução da relaxação Lagrangiana foi implementada no arquivo *lagrangean.cpp*, que utilizou o modelo do TSP fornecido pelo arquivo *tsp.cpp* e heurística Lagrangiana fornecida

pelo arquivo *lh.cpp*. O modelo linear para o kSTSP foi implementado o arquivo *kstsp.cpp* e a sua relaxação linear no arquivo *relaxed.cpp*. A relaxação Lagrangiana, o modelo linear e a sua relaxação linear fornecem arquivos de saída com informações da solução.

4.2. Instâncias

As instâncias foram geradas a partir do arquivo de coordenadas fornecido pelo professor, e possuem o formato a seguir, onde n é o número de vértices e k o fator de similaridade. A Tabela 4.2 apresenta um resumo das instâncias geradas.

$< n >$

$< k >$

$< coordenada\ x_1^1 > \quad < coordenada\ y_1^1 > \quad < coordenada\ x_1^2 > \quad < coordenada\ y_1^2 >$

$< coordenada\ x_2^1 > \quad < coordenada\ y_2^1 > \quad < coordenada\ x_2^2 > \quad < coordenada\ y_2^2 >$

...

$< coordenada\ x_n^1 > \quad < coordenada\ y_n^1 > \quad < coordenada\ x_n^2 > \quad < coordenada\ y_n^2 >$

4.3. Experimentos

As instâncias foram executadas exatamente um vez para todos os modelos: modelo de relaxação Lagrangiana, modelo linear e relaxação linear. O tempo de execução das instâncias foi limitado a 1800 segundos.

Para as instâncias 1, 4, 7 e 10, onde k era igual a 0, utilizou-se todos os multiplicadores de Lagrange como 0, com a intenção de não penalizar a função objetivo e obter, na primeira iteração, a solução ótima por meio da resolução dos TSP's. Para as demais instâncias inicializou-se os multiplicadores com o valor 4. Utilizou-se $\pi = 2$, com decaimento de 1% a cada iteração.

Como única condição de parada antes do tempo limite, utilizou-se o GAP entre o LB e o UB.

Instância	Vértices	K
1	100	0
2	100	50
3	100	100
4	150	0
5	150	75
6	150	150
7	200	0
8	200	100
9	200	200
10	250	0
11	250	125
12	250	250

Tabela 1: Tabela de instâncias.

5. Resultados e Análise

A Tabela 2 apresenta os resultados das execuções para o modelo linear apresentado na atividade pelo professor e a sua relaxação linear. Podemos observar que quando $k = 0$ o problema se torna mais fácil de resolver, pois os ciclos não têm obrigatoriedade de compartilhar arestas. Quando $k = \frac{|V|}{2}$ o problema se torna mais difícil de resolver, pois é nesse momento onde há a maior possibilidade de combinação de arestas. Dessa forma o algoritmo tem muitos nós para explorar, aumentando o seu tempo de convergência.

É possível observar que, em geral, o valor da relaxação linear está próximo do ótimo.

Na Tabela 3 encontramos os resultados para o modelo de relaxação lagrangiana com o método dos subgradientes. Os casos onde $k = 0$ foram trivialmente resolvidos por meio dos TSP's, ou seja, apenas juntando os melhores ciclos para cada um dos caixeiros. Para maiores valores de n, o LB ao término do tempo limite ficou mais distante do ótimo (apresentado na Tabela 2). Isso acontece devido ao fato de, quanto maior o valor de n, maior a demora para solucionar os TSP's. Com isso, menos iterações são feitas dentro do tempo limite, fazendo com que os multiplicadores de Lagrange sejam atualizados menos

Instância	Relaxação	UB	LB	Tempo (s)
1	1274	1630	1630	2.6
2	1881.76	2102	2102	23.9
3	3104	3463	3463	2.9
4	1583	1966	1966	10.3
5	2530.48	2748	2748	63.5
6	4331	4780	4780	11.0
7	1861	2308	2308	30.1
8	3044.58	3403	3403	1267.3
9	5414	6003	6003	25.1
10	2140	2597	2597	118.5
11	3588.43	10829	3936.7	1800
12	6319	6999	6999	293

Tabela 2: Resultados para o modelo linear a sua relaxação.

vezes. Dessa forma, eles ficam mais distantes do seu valor ótimo. Ainda assim, para as instâncias menores, os limitantes inferiores não ficaram muito distantes do valor ótimo.

Instância	LB	UB	Tempo (s)	Iterações
1	1630	5625	1	1
2	2053.31	4417	1800	919
3	3295.11	3703	1800	1032
4	1966	8590	4	1
5	2622.96	6834	1800	454
6	4465.83	5278	1800	655
7	2308	11775	9	1
8	2951.31	8976	1800	271
9	5444.92	7579	1800	394
10	2597	14668	30	1
11	3256.92	10884	1800	165
12	4850.25	10642	1800	163

Como piorou a relaxação linear?

Isso parece um problema de implementação.

Tabela 3: Resultados para o modelo de relaxação Lagrangiana.

Os limitantes superiores fornecidos pela heurística Lagrangiana ficaram extremamente distantes do ótimo. Isso pode ter sido um fator decisivo na não obtenção dos valores ótimos para os limitantes inferiores. No gráfico da Figura 1 podemos observar a evolução dos limitantes inferiores ao longo das iterações. Para as instâncias menores, como 2 e 6, notamos que eles rapidamente chegam próximo do ótimo, com cerca de 300 iterações, mas depois acabam estagnando. Caso os limitantes superiores se aproximassem para uma convergência, poderíamos ter obtido valores ótimos para os limitantes inferiores.

A instância 9 corrobora com a ideia de que para as instâncias maiores o avanço dos limitantes inferiores é mais lento, dando a impressão de que ainda após 400 iterações poderia aumentar um pouco mais antes de estagnar.

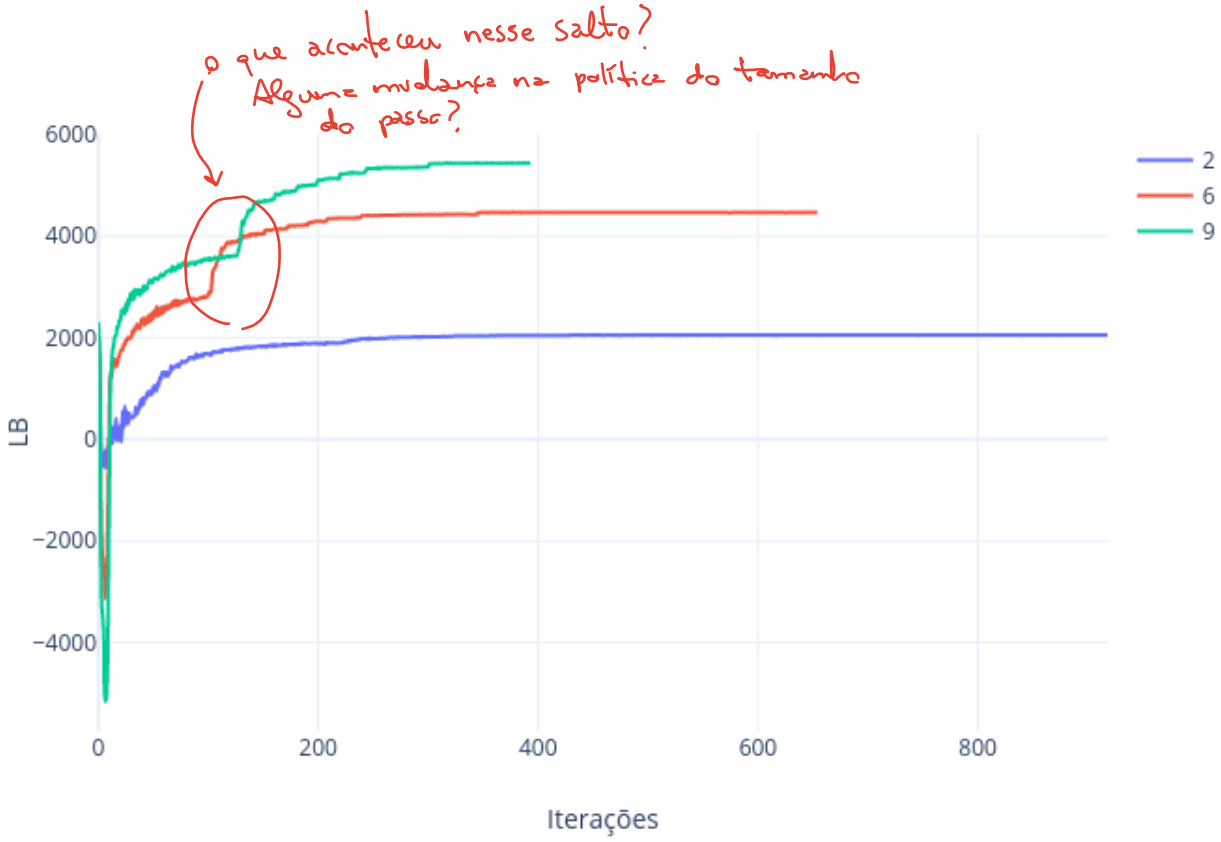


Figura 1: Evolução do LB para as instâncias 2, 6 e 9.

Submetido para disciplina MO824

Atividade 3 - Relaxação Lagrangiana do kTSP

Athyrson Machado Ribeiro RA 203963

Felipe Pereira RA 263808

Ieremies Vieira Da Fonseca Romero RA 217938

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: Propôr, modelar e resolver uma relaxação Lagrangiana do kSTSP através do método dos subgradientes que possa 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:

$$\sum_{e \in E} c_e^1 \; X_e^1 + \sum_{e \in E} c_e^2 \; X_e^2 + \lambda e \in E(D_e e \in E - c_e^1) \tag{2.1}$$

Sujeito a:

$$\sum_{e \in \delta(v)} X_e^1 = 2 \qquad \forall v \in V \tag{2.2}$$

$$\sum_{e \in \delta(v)} X_e^1 = 2 \qquad \forall v \in V \tag{2.3}$$

$$\sum_{e \in E(S)} X_e^1 \leq |S| - 1 \qquad \forall S \subseteq V, S \neq V, S \neq \emptyset \tag{2.4}$$

$$\sum_{e \in E(S)} X_e^2 \leq |S| - 1 \qquad \forall S \subseteq V, S \neq V, S \neq \emptyset \tag{2.5}$$

$$X_e^1 + X_e^2 \leq 2 \; D_e \qquad \forall e \in E \tag{2.6}$$

$$\sum_{e \in E} D_e \geq k \tag{2.7}$$

$$X_e^1, X_e^2, D_e \in \{0, 1\} \qquad \forall e \in E \tag{2.8}$$

2.3. Explicação das Restrições

- A função objetivo (2.1) é soma do custo de todas as arestas selecionadas.
- As restrições (2.2) e (2.3) 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 (2.4) e (2.5) 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 (2.6) garante que, se uma aresta foi escolhida para ser duplicada, então essa aresta aparecerá nos dois ciclos.
- A restrição (2.8) 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 (2.2) e (2.3): uma para cada vértice. Total: $2 \cdot |V|$;
- Restrições (2.4) e (2.5): uma para $S \in \mathcal{P}(V), S \neq V, S \neq \emptyset$. Total: $2 \cdot \left(2^{|V|} - 2\right)$;
- Restrições (2.6): uma para cada aresta. Total: $|E| = \frac{|V|^2 - |V|}{2}$ (já que o grafo é completo);
- Restrições 2.7: apenas uma. Total: 1;

Assim, o número total de restrições é:

$$T_r = 2 \cdot |V| + 2 \cdot \left(2^{|V|} - 2\right) + \frac{|V|^2 - |V|}{2} + 1 \tag{2.9}$$

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 atavés 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.

Faltou a apresentação e discução do dual Lagrangiano

2.5. Sub-Gradientes

A codificação dos sub-gradientes pode ser encontrada na função "subgradient"do arquivo lagrange.py

Seu pseudo-código corresponde à:

```
procedure calculateLagrangeWithSubGradients
    set Capitals K from input data
    Set Lagrange lagrange <- [ {e: 0 for e in dist[t]} for t in range(2) ]
    set Distance Dist from input data
    Set upper bound upper <- H(K,Dist)
    Initialize model gurobi-model <- gurobi(K, Dist, lagrange, upper)
    Set iterative variable i = 0

    While execution is not timed-out:
        Set sub-gradients subg <- subgradient(gurobi-model.vars, gurobi-
model.dup)
        Set step alpha <- passo(pi, upper, gurobi-model.objVal, subg)
        for t in {0,1}:
```

Qual foi a política adotada para ajustar o tamanho do passo?

```
            for e in subg[t]:
               Reset lagrange[t][e] <- max(lagrange[t][e] + alpha * subg[t][e], 0.0)
        Update model gurobi-model <- kstsp.gurobi(capitals, dist, lagrange, upper)
        Loop to next Capital

end calculateLagrangeWithSubGradients
```

2.6. Heurística

A heurística implementada no arquivo *heuristic.py* utiliza-se da ideia gulosa de melhor vizinho para o TSP comum. Nela, a cada interação adicionamos o vizinho mais próximo ao ciclo e mudamos para ele. Alteramos para que as decisões dos k primeiros vizinhos sejam iguais para os dois tuors e para isso, utilizamos o custo somado das arestas nos dois tuors para determinar a de menor custo. Dado que as k primeiras arestas sejam iguais entre os dois tuors, completamo-nos utilizando o mesmo algoritmo, entretanto para cada tuor separadamente.

Seu pseudo-código corresponde a:

```
procedure heuristic:
     Set not-visited list with all Capitals
     Set actual with first Capital
     Remove actual Capital from not-visited

     While there is a capital to be visited:
        Calculates distance to all not-visited Capitals
        Finds closest Capital
        Set actual to closest Capital
        Increments total distance total-dist += closest-capital-dist
        Remove actual Capital from not-visited
        Iterates to next element

    Return total-dist

end heuristic
```

3. Experimento Computacional

3.1. Configuração da Máquina

O problema foi executado num desktop: AMD Ryzen 1800x (16MB cache, 3.6GHz max boost 4.0GHz, 8 cores e 16 threads), 16GB DDR4-SDRAM, 240 GB SSD, Nvídia GeForce GTX 1060 com 6GB de vRam.

O sistema operacional foi o Windows 10 (versão 21H1), python 3.10.4 e Gurobi Optimizer v9.5.1[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.

4. Resultados

4.1. Resultados Obtidos

O programa foi construído para processar todas as instâncias e parâmetros pedidos no exercício. Abaixo podemos ver nas Tabelas 1 e Tabela 2 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 nas tabelas Tabela 1 e Tabela 2.

5. Análise e Observações Finais

Esta atividade possui resultados interessantes, pois possuímos dois métodos de processamento a comparar: Relaxação Linear e Método do Subgradiente. Não podemos deixar de citar que, dado o fato de termos realizado a Atividade 02 (conforme resultados da Tabela 3), também possuímos alguns dados de comparação do método Sem Relaxação em comparação ao método Com Relaxação, que usaremos apenas

Tabela 1: Experimentos com método do subgradiente com a heurística Lagrangiana

Instância		**kTSP**		
N° de vértices	k	Upper Bound	Lower Bound	Tempo (s)
100	0	2032	1630	1798.89s
100	50	3046	1515	1799.36s
100	100	3727	3042	1797.66s
150	0	2450	1966	1793.45s
150	75	3903	1979	1798.92s
150	150	5326	3747	1781.58s
200	0	2830	2308	1786.83s
200	100	4908	2283.44	1755.89s
200	200	6771	2321	1781.88s
250	0	3211	2104	1725.12s
250	125	5696	2597	1791.90s
250	250	7858	2513	1724.43s

como uma curiosidade e não como uma comparação formal, dado que foram feitos de maneira potencialmente diferentes, por grupos diferentes.

Em termos de tempo de execução, apenas para os métodos estritamente propostos para esta atividade, temos os métodos com Relaxação Linear em primeiro lugar e os métodos com Sub-Gradiente em segundo lugar. Podemos verificar que, quanto menor o número de vértices, maior é o custo-benefício das soluções com Relaxação Linear, dado que o tempo de execução é ordens de grandeza menor, enquanto que a performance fica dentro de um range de distância em torno de 10%.

Porém, quanto maior o número de vértices e, principalmente, quanto maior o grau de similaridade, maior fica o gap de desempenho entre as duas soluções, com um desempenho melhor para os métodos que utilizam o Sub-Gradiente.

Mas, de maneira geral, caso o problema possa ser resolvido sem as técnicas de relaxação, em um tempo de execução factível, os métodos Sem Relaxação possuem o melhor desempenho em termos de resultados obtidos em relação à função objetivo, o que já seria esperado pela teoria, de qualquer forma.

Podemos verificar que existe uma relação linear entre o resultado obtido e a relação Vértice-Similaridade na Relaxação Linear, conforme visto na Figura 1

Já o método utilizando os Sub-Gradientes possui uma característica interessante de possuir uma relação ligeiramente linear até certo ponto, conforme se aumenta a complexidade das quantidades de vértices e graus de similaridade, porém, após determinado ponto, este método entra em um platô de performance, diferentemente do método de Relaxação Linear, conforme podemos verificar na Figura 2

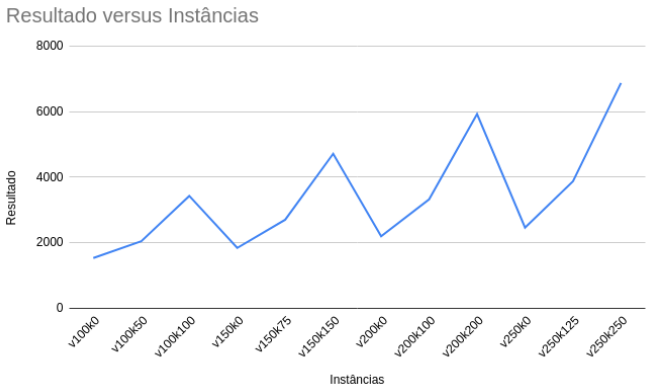


Figura 1: Resultados da Relaxação Linear

Figura 2: Resultados da relaxação Lagrangiana pelo método de Sub-Gradiente

isso sugere algum problema com o ajuste no tamanho do passo.

Tabela 2: Experimentos com método de Relaxação Linear com a heurística Lagrangiana

Instância		**kTSP**		
N° de vértices	k	Upper Bound	Lower Bound	Tempo (s)
100	0	2032	1529	0.02s
100	50	3046	2042	0.37s
100	100	3727	3028	0.27s
150	0	2450	1839	0.11s
150	75	3903	2702	0.75s
150	150	5326	4718	0.60s
200	0	2830	2193	0.22s
200	100	4908	3325	1.57s
200	200	6771	5930	1.22s
250	0	3211	2461	0.36s
250	125	5696	3879	2.45s
250	250	7858	6883	2.02s

Tabela 3: Resultados ótimos obtidos na Atividade 2. Todos com gap de 0% caso seja útil para a análise.

	$k = 0$	$k = \frac{v}{2}$	$k = v$
$v = 100$	1630	2102	3463
$v = 150$	1966	2748	4780
$v = 200$	2308	-	6003
$v = 250$	2916	-	6999

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.

Atividade 3 - MO824A/MC859A

João Victor Aquino Batista - RA: 199798
Adolfo Aires Schneider - RA: 234991

Este relatório tem como objetivo apresentar os resultados obtidos na atividade 3 da matéria MO824A/MC859A.

Definição Formal do Problema

O problema dos caixeiros viajantes *k*-semelhantes pode ser representado por um grafo não orientado G(V,E), onde V é o conjunto dos vértices e E é o conjunto das arestas, onde para cada aresta pertencente ao conjunto citado anteriormente existem dois custos c_e^1, c_e^2 pertencentes aos reais. Dado tal fato, queremos encontrar dois ciclos Hamiltonianos minimizando o custo total, de forma que pelo menos *k* arestas do grafo sejam visitadas por ambos os ciclos.

O fator *k* determina o quão similar são os ciclos, ou seja, se k = $|V|$, os dois ciclos contém, exatamente, as mesmas arestas; caso k = 0, então a solução é definida por qualquer par de ciclos Hamiltonianos.

Formulação Matemática

Além das variáveis do problema foram criadas as seguintes variáveis binárias de decisão, que serão necessárias para aplicação ao software GUROBI e as seguintes definições:

- x_e^k sendo a variável binária que representa se a aresta *e* está na rota do caixeiro *k* ou não;
- z_e sendo a variável binária que representa se a aresta e está na rota de ambos os caixeiros;
- $\delta(i)$ sendo o subconjunto de arestas que incidem no vértice *i*;
- $S \subset V$ sendo o subconjunto próprio de vértices;
- $E(S)$ sendo o conjunto das arestas cujos dois vértices terminais estão em *S*;

Tendo definido tal problema, pode-se formular um modelo de programação linear inteira com o objetivo de minimizar a expressão que representa a soma dos custos de todas as arestas presentes na solução:

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

obedecendo às seguintes restrições:

1. $\sum_{e \in \delta(i)} x_e^k = 2 \qquad \forall i \in V, \ \forall k \in \{1, 2\}$

2. $\sum_{e \in E(S)} x_e^k \leq |S| - 1 \quad \forall S \subset V, \forall k \in \{1,2\}$

3. $x_e^k \geq z_e \quad \forall e \in E, \forall k \in \{1,2\}$

4. $\sum_{e \in E(S)} z_e \geq k$

5. $x_e^k, z_e \in \{0,1\} \quad \forall e \in E, \forall k \in \{1,2\}$

A restrição (1) garante que cada vértice deve possuir duas arestas presentes no ciclo de cada um dos caixeiros. A restrição (2) garante que não haverão subciclos ilegais (cada caixeiro deve ser um único ciclo que visita todos os vértices do grafo). A restrição (3) garante que, se caso $z_e = 1$, então a aresta *e* estará contida no ciclo dos dois caixeiros. A restrição em (4) garante que pelo menos *k* arestas estejam no ciclo de ambos os caixeiros.

O conjunto de restrições escolhido para ser ~~realizado~~ dualizado foi o 4 e foi introduzido o multiplicador de lagrange $\boldsymbol{\lambda}$, resultando na seguinte função objetivo:

$$min \sum_{k \in \{1,2\}} \sum_{e \in E} c_e^k x_e^k + \boldsymbol{\lambda}(k - \sum_{e \in E} z_e)$$

Com isso o modelo resultante da relaxação lagrangeana é o seguinte.

$$min \sum_{k \in \{1,2\}} \sum_{e \in E} c_e^k x_e^k + \boldsymbol{\lambda}(k - \sum_{e \in E} z_e)$$

obedecendo às seguintes restrições:

6. $\sum_{e \in \delta(i)} x_e^k = 2 \quad \forall i \in V, \forall k \in \{1,2\}$

7. $\sum_{e \in E(S)} x_e^k \leq |S| - 1 \quad \forall S \subset V, \forall k \in \{1,2\}$

8. $x_e^k \geq z_e \quad \forall e \in E, \forall k \in \{1,2\}$

9. $x_e^k, z_e \in \{0,1\} \quad \forall e \in E, \forall k \in \{1,2\}$

Após a modelagem, foi executada a aplicação do método do subgradiente, objetivando encontrar o melhor limitante dual de forma iterativa, com passos na direção do subgradiente da função objetivo em relação a λ.

Implementação

Implementamos o modelo construído para o problema com a biblioteca gurobipy, implementação do software Gurobi para a linguagem de programação Python. Um pseudo-código em alto nível é apresentado abaixo, Algoritmo 1.

Método do Subgradiente

Algorithm 1 Método de Subgradientes

1: Inicialização do valor do multiplicador λ para a restrição relaxada.
2: **while** Tempo total for menor que tempo máximo *e* nenhum critério de parada for atingido **do**
3: Criar o modelo PLI relaxado com o valor ~~valor~~ atualizado de λ
4: Resolver o modelo criado e conseguir um Lower Bound LB
5: Utilizar a heurística lagrangiana vista em Algoritmo 2 para conseguir um Upper Bound UB
6: Obter o valor G de quanto a restrição foi violada
7: Calcular o passo com a formula $step = (2.0/it) * (UB - LB)/(G^2)$
8: Atualizar o valor do multiplicador com a formula $\lambda = max(0, \lambda + step * G)$
9:

Faltou explicar a motivação dessa estratégia

Sobre os critérios de parada utilizados, foram três, incluindo a otimalidade da solução, o limite de tempo e as violações nulas da restrição relaxada. A otimalidade da solução é obtida quando o limite superior é igual ao limite inferior. O terceiro caso acontece quando uma solução ótima do modelo dual é encontrada sem nenhuma violação da restrição relaxada, não afetando a função objetivo, ou seja, os multiplicadores não vão sofrer alterações. Nesse caso a solução dual é a solução ótima do modelo primal.

boa explicação

Heurística

Dado os ciclos halminonianos *A* e *B*, é escolhido um dos ciclos de forma arbitrária como sendo a primeira parte da solução *A* e desconsiderando o segundo ciclo *B*. Com isso, precisamos construir um novo ciclo *B2*, para substituir *B*, porém que seja factível. Para isso escolhemos as *k* arestas consecutivas de A que contém a menor soma dos custos e fixamos em *B2*. Com isso *B2* possui *K+1* vértices. Para tornar *B2* em um ciclo hamiltoniano válido é necessário o ciclo visitar os *N* vértices, isso é feito adicionando os vértices que faltam de forma gulosa no final de *B2*. O algoritmo é apresentado abaixo, Algoritmo 2.

Por que esse viés para o ciclo A? Não poderia ser o melhor de ambos? Além disso note que essa heurística descarta inteiramente a informação presente na rota B.

Algorithm 2 Heurística Lagrangiana

Require: Ciclos Hamiltonianos (cycleA, cycleB); k; matriz de adjacência com os custos das arestas (distCA, distCB)

```
if k == 0 then
    return cycleA, cycleB
end if
minCost = INF
newCicle = [ ]
for i = 1, 2, ... N − K + 1 do
    kCycle = cycleA[i: i + k + 1]
    kCycleCost = cost(kCycle)
    if kCycleCost < minCost then
        newCicle = kCycle
        minCost = kCycleCost
    end if
end for
while len(newCycle) < N do
    nodeToAppend = -1
    minEdgeCost = INF
    lastNode = newCycle[len(newCycle)-1]
    for v = 1, 2, ... N do
        if v not in newCycle AND distCB[lastNode, v] < minEdgeCost then
            nodeToAppend = v
            minEdgeCost = distCB[lastNode, v]
        end if
    end fornewCycle.append(nodeToAppend)
end while
    return cycleA, newCycle
```

Geração de Instâncias

As instâncias utilizadas foram as mesmas presentes na atividade anterior, totalizando 12 instâncias combinando quatro conjunto de vértices ($|V| = \{100, 150, 200, 250\}$) e três valores de similaridade ($k = \{0, |\frac{V}{2}|, |V|\}$).

Resultados

Na execução dos experimentos computacionais, foi utilizado um computador com 16GB de memória RAM, processador Intel® Core™ i5-9300H CPU @ 2.40 GHz × 8 e sistema operacional Windows 11.

Os resultados obtidos podem ser visualizados na tabela a seguir, sendo que as colunas "Cost (PL)" e "Time (PL)" são os resultados obtidos através da resolução do problema através

↳ Limitante inferior ou superior?

de programação linear, enquanto as outras são resultado da relaxação Lagrangiana. LB e UB representam o *lower bound* e o *upper bound*, respectivamente.

Instância	N	K	Cost (PL)	Time (PL)	LB	UB	Time
1	100	0	1.630,00	10.71	1.630,00	1.630,00	4,81
2	100	50	2.102,00	115.75	2.102,00	3.342,00	59,43
3	100	100	3.463,00	44,60	3.463,00	3.463,00	8,60
4	150	0	1.966,00	52.61	1.966,00	1.966,00	20,69
5	150	75	2.748,00	693.45	2.748,00	4.697,00	293,80
6	150	150	4.780,00	42.61	4.780,00	4.780,00	54,05
7	200	0	2.308,00	153.94	2.312,00	2.312,00	100,06
8	200	100	3390*	1800**	2.308,00	7.080,00	1.801,5**
9	200	200	6.003,00	49.27	6.003,00	6.003,00	157,70
10	250	0	2.597,00	540.69	2.599,00*	2.599,00*	400,73
11	250	125	3912*	1800**	2.597,00*	8.846,00*	1.802,3**
12	250	250	6.999,00	426.44	6.999,00	6.999,00	691,57

Tabela 1: Tabela com os dados de execução do modelo para cada conjunto de N e K iterados.
*Último valor exibido antes da interrupção pelo limite de tempo
**Execução não chegou ao fim e alcançou o tempo limite de 30 minutos de execução

Através da análise dos resultados podemos observar que o custo das soluções obtida através da programação linear, na maioria das vezes, é igual ou muito próxima da lower-bound obtida através da relaxação, o que indica que esta obteve um resultado fidedigno, porém com um tempo de execução inferior ao do método de PL (salvo alguns casos, como da instância 8 e 11 em que o limite de tempo total foi atingindo).

Também pode-se notar que, ao aumentarmos o números de vértices no grafo, o tempo e custo da solução tende a aumentar, já que mais vértices deverão ser "percorridos" para que um ciclo seja encontrado. O mesmo também ocorre com o fator k; quando este sobe de 0 para um valor que está dentro do intervalo $0 < k < V$, o custo e tempo da solução aumenta, já que temos um problema mais complexo, dado que o número de arestas em comum entre os dois ciclos aumenta. Entretanto, quando $k = V$, temos uma diminuição de custo e tempo, sendo que em alguns casos estes são menores do que quando $k = 0$. Isso ocorre pois os dois ciclos irão conter, exatamente, as mesmas arestas, logo a busca pelo segundo ciclo é mais direcionada.

MO824 - Tópicos em Otimização Combinatória

Relaxação Lagrangiana

Equipe 8

Gian Franco, Luiz Gustavo S. Aguiar, Pedro R. S. Mota

RAs: 234826, 240499, 185853

29 de Abril de 2022

Objetivo

O objetivo desta atividade consiste na aplicação da técnica de relaxação Lagrangiana, resolvida pelo método dos subgradientes, para a obtenção de limitantes primais e duais do problema dos caixeiros viajantes k-semelhantes (k-STSP).

Descrição do k-STSP

O kSTSP pode ser descrito da seguinte forma: seja um grafo não-orientado completo $G(V, E)$, onde V é o conjunto dos vértices e E é o conjunto das arestas, onde para cada aresta $e \in E$ há dois custos $c_e^1, c_e^2 : E \Rightarrow \mathbb{R}^+$, o objetivo do problema consiste em encontrar dois ciclos Hamiltonianos com custo total mínimo, de forma que pelo menos k arestas do grafo sejam visitadas por ambos os ciclos. O parâmetro k define a similaridade entre os ciclos, de modo que $k = |V|$ implica em uma solução onde os dois ciclos contém exatamente as mesmas $|V|$ arestas, enquanto $k = 0$ implica em uma solução onde os ciclos podem ser completamente distintos. O custo das arestas será sempre um inteiro positivo, pois será o teto da distância euclidiana.

Modelagem

Um modelo de programação linear inteira para o k-STSP é descrito a seguir:

$$MIN \sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k x_e^k \tag{1}$$

$$\sum_{e\in\delta(i)} x_e^k = 2,\ \forall i \in V,\ \forall k \in \{1,2\} \tag{2}$$

$$\sum_{e\in E(S)} x_e^k \leq |S| - 1,\ \forall S \subset V,\ \forall k \in \{1,2\} \tag{3}$$

$$x_e^k \geq z_e,\ \forall e \in E,\ \forall k \in \{1,2\} \tag{4}$$

$$\sum_{e\in E} z_e \geq k \tag{5}$$

$$x_e^k,\ z_e \in \{0,1\},\ \forall e \in E,\ \forall k \in \{1,2\} \tag{6}$$

Onde:

- x_e^k : Variável binária que diz se a aresta e foi utilizada no ciclo k;
- z_e : Variável binária que diz se a aresta e foi utilizada em ambos os ciclos;
- $\delta(v)$: conjunto de arestas incidentes no vértice v;
- S C V é um subconjunto próprio de vértices;
- $E(S)$ é o conjunto das arestas cujos dois vértices terminais estão em S

A função objetivo (1) minimiza o custo da solução, composto pela soma dos custos de todas as arestas presentes na solução. O conjunto de restrições (2) diz que cada vértice deve possuir duas arestas incidentes para o ciclo de cada um dos dois caixeiros. O conjunto de restrições (3) visa a eliminação de subciclos ilegais, ou seja, a rota de cada caixeiro deve corresponder a um único ciclo que visita todos os vértices. O conjunto de restrições (4) garante que, caso ze = 1, então a aresta e estará presente nos ciclos dos dois caixeiros. Finalmente, a restrição (5) força pelo menos k arestas em comum nos ciclos dos dois caixeiros.

Relaxação Lagrangiana

A relaxação Lagrangiana foi feita relaxando a restrição expressa na equação (4) e fazendo $k \geq 0$ na restrição da equação (5). Dessa forma o problema relaxado consiste em achar o melhor

ciclo hamiltoniano para cada caixeiro, sem a necessidade de que eles tenham uma similaridade mínima. Dessa forma, podemos formular um modelo para o problema do limitante inferior Lagrangiano, descrito a seguir:

$$MIN \sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k \, x_e^k + \lambda_k (z_e^k - x_e^k) \tag{7}$$

$$\sum_{e\in\delta(i)} x_e^k = 2, \, \forall i \in V, \, \forall k \in \{1,2\} \tag{8}$$

$$\sum_{e\in E(S)} x_e^k \leq |S| - 1, \, \forall S \subset V, \, \forall k \in \{1,2\} \tag{9}$$

$$\sum_{e\in E} z_e \geq k \tag{10}$$

$$x_e^k, \, z_e \in \{0,1\}, \, \forall e \in E, \, \forall k \in \{1,2\} \tag{11}$$

A partir desse problema, podemos formular o modelo do dual Lagrangiano:

$$MAX \; \lambda \; MIN \; x \sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k \, x_e^k + \lambda_k (z_e^k - x_e^k) \tag{12}$$

$$\sum_{e\in\delta(i)} x_e^k = 2, \, \forall i \in V, \, \forall k \in \{1,2\} \tag{13}$$

$$\sum_{e\in E(S)} x_e^k \leq |S| - 1, \, \forall S \subset V, \, \forall k \in \{1,2\} \tag{14}$$

$$\sum_{e\in E} z_e \geq k \tag{15}$$

$$x_e^k, \, z_e \in \{0,1\}, \, \forall e \in E, \, \forall k \in \{1,2\} \tag{16}$$

Heurística Lagrangiana

1. Encontre dois ciclos C_1 e C_2 de TSP.
2. Escolha o ciclo com menor custo para ser o ciclo base, suponha que nesse caso seja o C_1.
3. Construa um ciclo hamiltoniano C seguindo os seguintes passos:

(a) Insira as arestas duplicadas de C_1 e C_2 em C.

(b) Caso o número de arestas duplicadas ainda não seja k insira arestas factíveis(não forma ciclo sem ser hamiltoniano) de C_1 em C.

(c) Insira as de G em relação ao caixeiro 1 em C até que C seja um Hamiltoniano.

As arestas inseridas nos passos anteriores são escolhidas por critério de menor custo. No momento em que C for um ciclo hamiltoniano basta retornar $W(C) + W(C_1)$ onde $W(X)$ é o custo do ciclo X.

Esse passo precisa ser melhor explicado

Realizando o algoritmo descrito anteriormente nós conseguimos encontrar o ciclo desejado com pelo menos k arestas compartilhadas mas note que isto é uma heurística logo não temos garantia de resposta ótima.

Resultados

Método dos subgradientes?
Política de ajuste do tamanho do passo?
Valor inicial de λ?

Implementamos o modelo descrito com o solver Gurobi na versão 9.5.1 utilizando a linguagem Python utilizando a versão 3.8.9, numa máquina com o processador Intel Core i7 com 2.60 GHz, 16GB de memória RAM e sistema operacional macOS Monterey 12.3.1.

O código implementado utiliza-se das coordenadas fornecidas e executa o modelo um total de 12 vezes, variando entre as possíveis combinações de V = {100, 150, 200, 250} e K = {0, V/2, V}.

Table 1: Métricas do modelo para as instâncias citadas no formato Heurística lagrangiana (limitante inferior, limitante superior, tempo de execução) e Relaxação linear (limitante inferior, limitante superior, tempo de execução.

		$k = 0$	$k = \frac{v}{2}$	$k = v$
v = 100	h. lagrangiana	(2528.4, 2531.0, 1.87)	(5343.7, 5355.0, 1.37)	(7442.2, 7460.0, 1.24)
	r. linear	(1670.0, 2531.0, 0.81)	(1670.0, 5355.0, 0.92)	(1670.0, 7460.0, 0.83)
v = 150	h. lagrangiana	(3084.5, 3088.0, 5.91)	(6254.9, 6269.0, 7.28)	(9414.3, 9439.0, 7.41)
	r. linear	(2026.0, 3088.0, 5.06)	(2026.0, 6269.0, 4.97)	(2026.0, 9439.0, 5.18)
v = 200	h. lagrangiana	(3757.2, 3762.0, 13.32)	(9087.7, 9111.0,14.99)	(13364.8, 13403.0, 14.99)
	r. linear	(2388.0, 3762.0, 6.93)	(2388.0, 9111.0,7.10)	(2388.0, 13403.0, 7.55)
v = 250	h. lagrangiana	(4149.8, 4155.0, 41.29)	(9781.8, 9807.0, 1:48.16)	(14739.1, 14782.0, 39.07)
	r. linear	(2697.0, 4155.0, 30.21)	(2697.0, 9807.0, 29.91)	(2697.0, 14782.0, 29.81)

O tempo da RL reflete todas as iterações do subgradiente?

Análise

Os resultados apresentados mostram que ao final das iterações do subgradiente, a relaxação lagrangiana obteve soluções com custo alto. Isso acontece por conta da heurística usada, que encontra uma solução factível e não necessariamente ótima. Apesar disso, os valores de upper bound e lower bound para o método do subgradiente com a heurística lagrangiana, são muito mais próximos do que os valores da relaxação linear. Essa diminuição do GAP entre os bounds, mostra que a relaxação lagrangiana teve efeito.

Faltou discussão qualidades dos limitantes para diferentes classes de instâncias.

Como os limitantes Lagrangianos convergiram ao longo das iterações do método de subgradientes? Lento, rápido?

Atividade 3

Grupo 5

Lucas Costa De Oliveira RA: 182410, Lucas Pedroso Cantanhede RA: 202058,
Lucas De Oliveira Silva RA: 220715

29 de Abril de 2022
MO824A/MC859A – Tópicos em Otimização Combinatória

Descrição do problema?

Modelo Matemático

Nos baseamos no modelo do kSTSP presente no enunciado da atividade para fazer a relaxação lagrangeana. Optamos por relaxar a restrição 5, ou seja, $\sum_{e \in E} z_e \geq K$ retirando ela do conjunto de restrições. Desse modo não garantimos uma similaridade k ao terminar de executar, para isso vamos aplicar a heurística lagrangeana.

Após a relaxação, usando o modelo do enunciado como base, temos a seguinte formulação, com as variáveis:

x_e^k = indica se a aresta *e* foi utilizada no caminho k

z_e = indica que a aresta *e* foi utilizada nos 2 caminhos

E utilizamos as seguintes constantes:

k = número de arestas presentes nos 2 caminhos

c_e^k = custo da aresta e no caminho k

V = conjunto de vértices do grafo

E = conjunto de arestas do grafo

S $\subset$ V = é um subconjunto próprio de vértices

$\delta(i)$ = conjunto de arestas que incide no vértice $i \in V$

$E(S)$= conjunto de arestas cujos 2 vértices terminais estão em S

Como objetivo queremos minimizar a função dada por:

$$MIN \sum_{e \in \{1,2\}} \sum_{e \in E} x_e^k c_e^k - \lambda\left(\sum_{e \in E} z_e - k\right)$$

Com as seguintes restrições:

$$(1) \sum_{e \in \delta(i)} x_e^k = 2 \ \forall i \in V, \forall k \in \{1,2\}$$

$$(2) \sum_{e \in E(S)} x_e^k \leq |S| - 1 \ \forall S \subset V, \forall k \in \{1,2\}$$

$$(3)\ x_e^k \geq z_e \ \forall e \in E, \forall k \in \{1,2\}$$

$$(4)\ x_e^k, z_e \in \{0,1\}, \forall k \in \{1,2\}$$

Que podem ser traduzidas como:

(1) A soma da quantidade de arestas incidentes no vértice i deve ser igual a 2 para garantir que o vértice é visitado apenas 1 vez no caminho k, ~~uma aresta entra e uma aresta sai do vértice.~~

(2) A soma da quantidade de arestas com os 2 vértices terminais em S, ou seja, no ciclo k, deve ser igual ou menor ao número de vértices presentes no ciclo k menos 1 para garantir que eliminaremos subciclos ilegais ao limitar o número de arestas presentes ~~num~~ em um subconjunto de vértices

(3) A variável que controla se uma aresta está no caminho k deve ser maior ou igual a variável que controla se a aresta está em ambos os caminhos, desse modo se uma aresta está em ambos os caminhos o seu valor de z deve ser 1 e o de ambos x também.
(4) Restrição de integralidade para nossas variáveis.

Método do subgradiente

Ao relaxar nossa formulação adicionamos um multiplicador λ em nossa função de minimização. Para resolver essa nossa nova formulação necessitamos encontrar o melhor valor possível para λ, esse problema é chamado de Dual Lagrangeano. Para resolver o problema do Dual Lagrangeano utilizamos o método do subgradiente. Para utilizar o método necessitamos de um limite superior, para o cálculo desse limite superior utilizamos uma simples heurística: considerando que ambos os ciclos são dados pela sequência de nós 1, 2, …, n calculamos a soma de ambos os custos para as arestas presentes. Partindo do nosso problema original temos um novo problema em que cada variável z_e será multiplicada por uma constante $\lambda_i \geq 0$. Representaremos o valor do limite inferior da iteração atual como LB e o valor do limite superior calculado como UB, temos uma variável π_i iniciada em 2 que será atualizada para garantir a convergência do algoritmo. Desse modo temos o seguinte algoritmo:

(1) $\lambda_0 = 0, \pi_0 = 2$

(2) Enquanto |UB - LB| e π_i não forem suficientemente pequenos:

(3) Resolvo o LLBF atual com o valor de λ_i e obtenho um novo LB

(4) $g_i = k - \sum_{e \epsilon E} z_e$ // Cômputo do subgradiente

(5) $\alpha_i = \pi_i$ (UB-LB)/$\sum g_i^2$ // Atualização dos valores de acordo com os limites atuais

(6) $\lambda_{n+1} = \max(0, \lambda_n + \text{alpha} \cdot g_i)$ //Atualizo meus multiplicadores lagrangeanos

(7) $\pi_{+1i} = \pi_i \cdot 0{,}9$ // Garantindo convergência

decaimento muito rápido

Ao fim desse algoritmo devemos ter os melhores valores possíveis para λ e um resultado de LB pois resolvemos o LBFF conforme calculamos os novos λ .

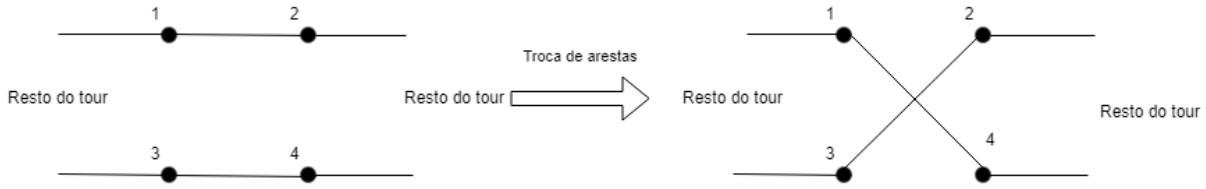
seria melhor aplicar a redução caso o LB piore

Heuristica Lagrangeana

Com a relaxação em nossa função de minimização retiramos a restrição característica do kSTPS que é o fator de similaridade k. Portanto, ao terminar a execução do algoritmo podemos ter um valor arbitrário para k, não necessariamente o desejado. Nos baseamos no 2-opt para implementar a heurística, mas a condição de troca não é o custo, é se a similaridade irá aumentar. Como podemos ver na imagem abaixo a simples troca de arestas nos mantém com um ciclo, portanto se essa troca aumenta a similaridade ou a mantém a executamos pois pode ser que o ciclo atual não nos permita uma troca em que a similaridade é aumentada, portanto realizamos uma troca em que a similaridade é

Gostei da heurística. Uma sugestão é também olhar o custo como critério secundário, ou seja, se dois movimentos melhoram K, escolha aquele que mais reduzir o custo

mantida analisamos novamente se com o novo ciclo existe uma troca em que a similaridade é aumentada.



Essa heurística tem como objetivo corrigir esse valor de k, vamos usar o termo similaridade para representar k. Seja t1 o primeiro tour e t2 o segundo tour, e(i,j) a aresta que liga os vértices i e j, ir (jr) e il (jl) sejam os vizinhos da direita e esquerda de i (j) respectivamente,

itálicos

(1) Para cada i em V
(2) Para cada j em V
(3) Se $e(i,j) \in t1$ e $e(i,j) \notin t2$
(4) sim = $1 + e(il, jr) \in t1 - e(i,il) \in t1 - e(j,jr) \in t1$
(5) Se sim ≥ 0: adicione e(i, j), e(il, jr) em t2, remova e(i, il), e(j, jr) em t2 e aumenta a similaridade em sim
(6) sim = $1 + e(ir, jl) \in t1 - e(i,ir) \in t1 - e(j,jl) \in t1$
(7) Se sim ≥ 0: adicione e(i,j), e(ir,jl) em t2, remova e(i, ir), e(j, jl) em t2 e aumenta a similaridade em sim

Resultado

Implementamos o modelo descrito numa máquina com o processador Intel Core i5-7300HQ com 2.50 GHz, 8GB de memória RAM e sistema operacional Pop! OS 21.10 e kernel 5.16.11. utilizamos o solver Gurobi na versão 9.5.1 utilizando a linguagem C++ compilador g++ 11.2.0 com as opções:

model.set(GRB IntParam MIPFocus, GRB MIPFOCUS FEASIBILITY);
model.set(GRB IntParam Cuts, GRB CUTS AGGRESSIVE);
model.set(GRB IntParam Presolve, GRB PRESOLVE AGGRESSIVE);

Métodos Subgradiente:

Instância #	Vértices (|v|) e Similaridade (k)	UB(Limitante Superior)	LB(Limitante Inferior)	Tempo de Execução (segundos)	λ Final	N^o Iterações do subgradiente
1	100 e 0	1630	1630	6,4	0,0	1
2	100 e 50	2102	2102	43,8	16,3	3
3	100 e 100	3463,0	3463,0	9,3	120,6	2
4	150 e 0	1966	1966	23,6	0,0	1

5	150 e 75	2748	2748	601,9	17,0	10
6	150 e 150	4780	4780	27,3	135,2	2
7	200 e 0	2308	2308	116,1	0,0	1
8	200 e 100	4611	3375,5	1800,0	17,7	3
9	200 e 200	6003	6003	139,7	150,9	2
10	250 e 0	2597	2597	464,0	0,0	1
11	250 e 125	6350	4807,5	1800,0	11,0	3
12	250 e 250	6999	6999	568,2	156,5	2

PLI da Seção 2:

Instância #	Vértices (|v|) e Similaridade (k)	Relaxação Linear	UB (Limitante Superior)	LB (Limitante Inferior)	Tempo de Execução (segundos)
1	100 e 0	1529	1630	1630	5,4
2	100 e 50	1529	2102	2102	64,2
3	100 e 100	1529	3463	3463	8,1
4	150 e 0	1839	1966	1966	14,2
5	150 e 75	1839	2748	2748	241,0
6	150 e 150	1839	4780	4780	29,9
7	200 e 0	2193	2308	2308	119,3
8	200 e 100	2193	3458	3390	1800,0
9	200 e 200	2193	6003	6003	152,6
10	250 e 0	2461	2597	2597	191,0
11	250 e 125	2461	-	3920	1800,0
12	250 e 250	2461	6999	6999	688,3

Análise

Um fato importante que observamos em nossos teste foi o valor da relaxação linear apenas ser alterado quando o valor |v| era alterado, sendo praticamente independente do valor de k. Para explicar esse comportamento cremos que, como ao aplicar a relaxação linear as variáveis z e x estão no intervalo [0,1], a solução ótima apenas depende de |v| pois, como o x pode ser muito pequeno, caso adicionemos mais arestas na solução a multiplicação x*c na função que queremos minimizar não levará em conta o valor de c se x for suficientemente pequeno. Desse modo podemos ir adicionando arestas sem considerar o custo real delas, estaríamos pegando apenas uma pequena porcentagem da aresta e considerando ela na solução. Do ponto de vista gráfico podemos pensar que ao alterar o valor de k o plano de solução é alterado, porém sempre irá existir um ponto ótimo para um valor de |v| que estará presente em todos os planos possíveis quando alteramos k.

Outro detalhe interessante é notar o valor final de λ para cada instância. Para todas as instâncias com k = 0 temos λ = 0, o que faz sentido pois a restrição que optamos por relaxar não limitava o problema em nenhum momento quando k=0, tornando a penalidade não necessária. Outro detalhe importante é que os valores de λ são todos positivos pois a expressão que ele multiplica é sempre menor que 0, desse modo qualquer valor $\lambda \neq 0$ irá adicionar uma penalidade. Também é notável que o parâmetro lambda cresce junto com o aumento da similaridade k, isso se deve a presença do termo λk positivo que "obriga" que o termo $-\sum_{e \in E} z_e$ seja suficientemente negativo.

Com o método da relaxação lagrangeana e heurística foi possível gerar soluções factíveis para as instâncias em que no trabalho anterior não foi possível definir UBs. Pois a condição de similaridade é uma restrição complicadora, e sem ela o modelo converge mais rápido. Além disso, a utilização de uma heurística também acelerou a busca.

Uma possível melhoria seria a aplicação de 2-OPT nas soluções inteiras da relaxação, para acelerar o método de branch-and-cut e portanto a obtenção de limitantes duais.

Além disso, podemos fazer algumas observações a respeito dos limitantes inferiores e superiores obtidos no método do subgradiente e do PLI. Neles, é possível observar que todas as instâncias que não deram timeout chegaram a um mesmo valor para o limitante inferior e superior, o que significa que encontramos a solução ótima. Já nas instâncias com timeout, apesar de não terem chego nesse mesmo resultado, é possível notar que os limitantes encontrados aparentam estar convergindo, o que nos faz acreditar que também chegariam ao valor ótimo caso a execução fosse feita até o final.

Atividade 3

Aplicação da Técnica de Relaxação Lagrangiana

Grupo 7

Matheus Cezar Nunes - RA: 203373
Rebecca Moreira Messias - RA: 186416
Vinicius da Silva - RA: 206734

Abril de 2022

MC859/MO824

1 Objetivo

Nessa atividade será aplicado a técnica de relaxação Lagrangiana, resolvida pelo método dos subgradientes, para a obtenção de limitantes primais e duais de um problema de programação linear inteira [2]. A solução será desenvolvida com o software Gurobi [1].

O problema que será tratado nessa atividade é do k-TSP, uma variação do problema do TSP [3] (problema do caixeiro viajante) e o mesmo trablhado na atividade anterior. Ele será explicado melhor na próxima sessão.

As próximas sessões consistem na descrição do problema, explicação do desenvolvimento da soução e nas conclusões obtidas.

2 Descrição do problema

O problema dos caixeiros viajantes k-semelhantes (k-similar travelling salesmen problem, kSTSP) pode ser descrito da seguinte forma. Seja um grafo não-orientado completo $G(V,E)$, onde V é o conjunto dos vértices e E é o conjunto das arestas. Em cada aresta $e \in E$ há dois custos $c_e^1, c_e^2 : E \to \mathbb{R}^+$. O objetivo do problema consiste em encontrar dois ciclos Hamiltonianos com custo total mínimo, tal que pelo menos k arestas do grafo sejam visitadas por ambos os ciclos. O parâmetro k define a similaridade entre os ciclos, de modo que $k = |V|$ implica que uma solução factível corresponda a dois ciclos Hamiltonianos contendo as mesmas $|V|$ arestas, enquanto $k = 0$ resulta em uma solução factível ser qualquer par de ciclos Hamiltonianos.

Um modelo de programação linear inteira para o kSTSP é fornecido a seguir:

$$MIN \quad \sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k x_e^k \tag{1}$$

$$s.t. \quad \sum_{e\in\delta(i)} x_e^k = 2 \qquad \forall i \in V, \forall k \in \{1,2\} \tag{2}$$

$$\sum_{e\in E(S)} x_e^k \leq |S| - 1 \qquad \forall S \subset V, \forall k \in \{1,2\} \tag{3}$$

$$x_e^k \geq z_e \qquad \forall e \in E, \forall k \in \{1,2\} \tag{4}$$

$$\sum_{e\in E} z_e \geq k \tag{5}$$

$$x_e^k, z_e \in \{0,1\} \qquad \forall e \in E, \forall k \in \{1,2\} \tag{6}$$

Onde:

- x_e^k é uma variável de decisão binária associada à presença ($x_e^k = 1$) ou não ($x_e^k = 0$) da aresta e na rota do caixeiro k;
- z_e é uma variável de decisão binária associada à presença ($z_e = 1$) ou não ($z_e = 0$) da aresta e nas rotas de ambos caixeiros;
- $\delta(i)$ é o conjunto de arestas que incidem no vértice i;
- $S \subset V$ é um subconjunto próprio de vértices;
- $E(S)$ é o conjunto das arestas cujos dois vértices terminais estão em S.

A função objetivo (1) minimiza o custo da solução, composto pela soma dos custos de todas as arestas presentes na solução. O conjunto de restrições (2) diz que cada vértice deve possuir duas arestas incidentes para o ciclo de cada um dos dois caixeiros. O conjunto de restrições (3) visa a eliminação de subciclos ilegais, ou seja, a rota de cada caixeiro deve corresponder a um único ciclo que visita todos os vértices. O conjunto de restrições (4) garante que, caso $z_e = 1$, então a aresta e estará presente nos ciclos dos dois caixeiros. Finalmente, a restrição (3.1) força pelo menos k arestas em comum nos ciclos dos dois caixeiros.

3 Desenvolvimento

3.1 Formulação Matemática

A escolha do conjunto de restrições a serem relaxadas foi a seguinte:

$$x_e^k \geq z_e \qquad \forall e \in E, \forall k \in \{1, 2\} \quad (4)$$

Como visto na atividade anterior, o problema dos caixeiros viajantes k-semelhantes apresenta maior dificuldade computacional quando necessário o compartilhamento de arestas entre os dois caxeiros, logo será este conjunto que será relaxado e terá uma solução heurística para encontrar um *upper bound*.

Uma possível formulação da relaxação Lagrangiana, mais especificamente o *Lagrangean dual*, ou (*LD*), é fornecido a seguir de acordo com a escolha do conjunto de restrições a serem relaxadas:

$$\max_{\lambda \geq 0} \left\{ \begin{array}{lll} MIN \quad Z_{LB} = \sum_{k \in \{1,2\}} \sum_{e \in E} c_e^k x_e^k + \lambda'(z_e - x_e^k) & & \\ s.t. & \sum_{e \in \delta(i)} x_e^k = 2 & \forall i \in V, \forall k \in \{1,2\} \\ & \sum_{e \in E(S)} x_e^k \leq |S| - 1 & \forall S \subset V, \forall k \in \{1,2\} \\ & \sum_{e \in E} z_e \geq k & \\ & x_e^k, z_e \in \{0,1\} & \forall e \in E, \forall k \in \{1,2\} \end{array} \right\}$$

Rearranjando algébricamente os termos da função objetivo:

$$para \quad \lambda_i \geq 0, (1, ..., k \times |E|) \tag{7}$$

$$\min_{LB} = \sum_{k \in \{1,2\}} \sum_{e \in E} c_e^k x_e^k + \sum_{k \in \{1,2\}} \sum_{e \in E} \lambda_e^k (z_e - x_e^k) \tag{8}$$

$$\min_{LB} = \sum_{k \in \{1,2\}} \sum_{e \in E} c_e^k x_e^k + \lambda_e^k (z_e - x_e^k) \tag{9}$$

$$\min_{LB} = \sum_{k \in \{1,2\}} \sum_{e \in E} \boldsymbol{C}_e^k x_e^k + \sum_{k \in \{1,2\}} \sum_{e \in E} \lambda_e^k z_e \tag{10}$$

$$onde \quad \boldsymbol{C}_e^k = c_e^k - \lambda_e^k \tag{11}$$

Onde temos os custos lagrangianos das arestas e e caxeiro viajante k denotados por: $\boldsymbol{C}_e^k$.

3.2 Geração das Instâncias

Nenhum experimento foi executado.

3.3 Método do subgradiente

Algorithm 1: Método do Subgradiente

Input :

- Z_{LB} obtido através da relaxação: lb.
- Z_{UB} obtido através da heurística lagrangiana: ub.
- lambdas inicializados $\lambda_i = 0, i \in \{0, .., k \times |E|\} : \lambda$
- Conjunto de variáveis z_e obtidos no cálculo do lower bound
- Conjunto de variáveis x_e^k obtidos no cálculo do lower bound
- O valor atual de π (quando k=0, inicializamos $\pi = 2$): π.

Output: $\lambda_i \in \{0, .., k \times |E|\}$

$new\_\lambda = [\,]$;
$g = [\,]$;
$g\_sum\_squared = 0$;
while $i < len(\lambda)$ **do**
    $g_i \leftarrow 2 \times z_i - (x_i^1 + x_i^2)$;
    $g \leftarrow g_i$;
    $g\_sum\_squared \leftarrow g\_sum\_squared + (g_i)^2$;
    $i \leftarrow i + 1$;
end
$\alpha \leftarrow \pi \times \frac{(Z_{UB} - Z_{LB})}{g\_sum\_squared}$;
$\pi \leftarrow \pi \times 0.99$;
while $i < len(g_s)$ **do**
    $new\_\lambda_i \leftarrow \max(0, \lambda_i + \alpha \times g_i)$;
    $new\_\lambda \leftarrow new\_\lambda_i$;
end
return new_λ;

3.4 Heurística Lagrangiana

Para obter um limite superior (primal) para o problema, a princípio, foi utilizado uma heurística muito ingênua. Lembrando que a restrição 4 foi relaxada, a heurística deve garantir a semelhança mínima entre as arestas optadas por ambos os viajantes. Outro ponto importante é que um parâmetro de semelhança é tratado como um requisito mínimo, ou seja, o parâmetro $\frac{V}{2}$, por exemplo, representa no mínimo $\frac{V}{2}$ arestas semelhantes, mas pode conter entre $\frac{V}{2} \leq x \leq V$.

Dado isso, a heurística valida se a solução encontrada respeita o critério de semelhança, caso respeite, não há mudança a ser feita e é declarado otimalidade pois foi alcançado $Z_{LB} = Z_{UB}$. Caso contrário, a heurística copia o tour de um dois caxeiros para outro cuja soma dos custos dos dois caxeiros seja menor, garantindo uma similaridade de $k = |V|$.

Abaixo segue o pseudocódigo da implementação:

Poderia pelo menos tentar fazer algumas melhorias com 2-opt. Note que sua heurística está descartando por completo a informação de uma das rotas.

Algorithm 2: Heurística Lagrangiana

Input :

- Duas listas representando os tours de cada caxeiro: $tour1$ e $tour2$.
- Duas matrizes representando o custo de cada aresta para cada caxeiro: $costs1$ e $costs2$.
- Critério de semelhança: k.

Output: L_{UB} representado pelo tour de cada caxeiro

```
similarity ← 0;
while e in tour1 do
    if e in tour2 then
        similarity ← similarity + 1;
    end
end
if k ≤ similarity then
    return (tour1, tour2);                                  // optimality
else
    tour1_cost ← get_cost(tour1, cost1) ;
    tour2_cost ← get_cost(tour2, cost2) ;
    inverted_cost_for_tour1 ← get_cost(tour1, cost2) ;
    inverted_cost_for_tour2 ← get_cost(tour2, cost1) ;
    if (tour1_cost + inverted_cost_for_tour1) ≤ (tour2_cost + inverted_cost_for_tour2)
     then
        return (tour1, tour1) ;     // Better use tour1 for second traveler
    else
        return (tour2, tour2) ;      // Better use tour2 for first traveler
    end
end
```

4 Análise

Nenhum experimento foi executado.

Referências

[1] Gurobi documentation. https://www.gurobi.com/documentation/9.5/.

[2] Fábio Luiz Usberti and Celso Cavellucci. Integer programming.

[3] Wikipedia contributors. Travelling salesman problem — wikipédia, a enciclopédia livre. https://en.wikipedia.org/w/index.php?title=Travelling_salesman_problem&oldid=1080774476, 2022. [Online; accessed 9-April-2022].

Excelente, parabéns!

Atividade 3 - Técnica de Relaxação Lagrangiana para o kSTSP

Lucas Guesser Targino da Silva - RA: 203534
Renan Fernando Franco da Silva - RA: 223989

1 de maio 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. $c^k : E \to \mathbb{R}_+$, $\forall k \in \{1, 2\}$, duas funções custo nos vértices;
   (a) dada uma aresta e, escrevemos $c^k(e) = c^k_e$;
3. σ: parâmetro de similaridade de ciclos;

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

Ótimo! Único grupo que notou esse abuso de notação!

O nome kSTPS significa *k-similar Travelling Salesman Problem*. O k é o parâmetro de similaridade σ acima, convenientemente renomeado para que não houvesse confusão com o k utilizado para indexar cada um dos ciclos (mais sobre isso na Seção 2).

2 Modelo Matemático

Nessa seção, será apresentada a formulação do problema utilizando Programação Linear Inteira. Esse não foi resolvido diretamente (já que não é o intuito da atividade), mas foi utilizado para derivar uma Relaxação Lagrangiana (Seção 3), que foi modelo de fato implementado e resolvido.

2.1 Variáveis de Decisão

- x^k_e: presença da aresta e no ciclo k;
- z_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:

$$\sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k \; x_e^k \tag{1}$$

Sujeito a:

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

$$\sum_{e\in E(S)} x_e^k \leqslant |S| - 1 \qquad \forall S \subsetneq V, S \neq \emptyset, \forall k \in \{1,2\} \tag{3}$$

$$x_e^k \geqslant z_e \qquad \forall e \in E, \forall k \in \{1,2\} \tag{4}$$

$$\sum_{e\in E} z_e \geqslant \sigma \tag{5}$$

$$x_e^k, z_e \in \{0,1\} \qquad \forall e \in E, \forall k \in \{1,2\} \tag{6}$$

2.3 Explicação das Restrições

- A função objetivo (1) é soma do custo de todas as arestas selecionadas em todos os ciclos.
- A restrição (2) garante que a quantidade de arestas incidentes em todos os vértices seja 2. 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").
- A restrição (3) garante que não existam subciclos nos ciclos. Nessa restrição, S é um subconjunto próprio e não-vazio dos vértices do problema. A expressão $E(S)$ é o conjunto das arestas cujos ambos os vértices estão em S.
- A restrição (4) garante que, se uma aresta foi escolhida para ser duplicada, então essa aresta aparecerá nos dois ciclos.
- A restrição (5) garante que pelo menos σ arestas serão escolhidas para serem duplicadas.
- A restrição (6) 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

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

Assim, o número total de restrições é:

$$T_r = 2 \cdot |V| + 2 \cdot \left(2^{|V|} - 2\right) + 2 \cdot \frac{|V|^2 - |V|}{2} + 1 \tag{7}$$

Assim:

$$T_r \in \mathcal{O}\left(2^{|V|}\right) \tag{8}$$

Note que há um número exponencial de restrições.

3 Relaxação Lagrangiana

3.1 Escolha da Restrição a ser Relaxada

A técnica de Relaxação Lagrangiana consiste em escolher uma ou mais restrições para relaxar. Tal escolha visa remover as restrições que fazem do problem "difícil de ser resolvido" [1].

3.1.1 Análise das Restrições

- Restrição 2: não parece ser uma restrição difícil;
- Restrição 3: essa restrição é de fato difícil. Entretanto, há um número exponencial delas, de forma que é inviável relaxá-la. Além disso, há técnicas que lidam bem com ela[1];
- Restrição 4: no trabalho anterior, notou-se que o fator de similaridade causa bastante impacto no tempo computacional Figura 1. Isso significa que ele deixa o problema difícil, sendo então um bom candidato à relaxação;
- Restrição 5: essa restrição tem efeitos bem parecidos com a Restrição 4. Ela representa, entretanto, apenas uma equação de restrição, não sendo assim interessante para a relaxação;
- Restrição 6: essa restrição faz parte do modelo relaxado;

Portanto, a restrição escolhida para ser relaxada é a Restrição 4.

3.2 Modelo Matemático da Relaxação Lagrangiana

$$\max_{\lambda} \left\{ \min_{x,z} \sum_{k \in \{1,2\}} \sum_{e \in E} \left(\mathcal{C}_e^k \; x_e^k + \lambda_e^k z_e \right) \right\} \tag{9}$$

Em que:

$$\mathcal{C}_e^k = c_e^k - \lambda_e^k \tag{10}$$

Sujeito às restrições 2, 3, e 5 do problema original (Subseção 2.2), e a restrições de domínio:

$$0 \leqslant x_e^k, z_e \leqslant 1 \qquad \forall e \in E, \forall k \in \{1,2\} \tag{11}$$

$$\lambda_e^k \geqslant 0 \qquad \forall e \in E, \forall k \in \{1,2\} \tag{12}$$

Note que esse problema pode ser resolvido quebrando-o em 2 TSP e um problema para resolver as arestas a serem duplicadas z_e. Tais detalhes são melhor descritos nas próximas subsubseções.

[1] *Lazy constraints* por exemplo, abordado no trabalho anterior.

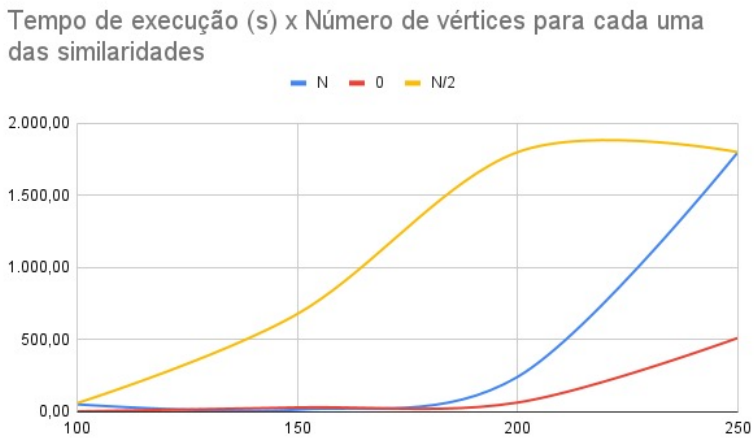


Figura 1: Tempo de execução dos experimentos do trabalho anterior.

3.2.1 Problema da escolha de ciclos derivado da Relaxação Lagrangiana

Dado $\mathcal{C}$, minimizar:

$$\sum_{k\in\{1,2\}} \sum_{e\in E} \mathcal{C}_e^k \; x_e^k \tag{13}$$

Sujeito às restrições 2, 3, e 6. Note que esse problema é o TSP.

3.2.2 Problema da escolha das arestas similares derivado da Relaxação Lagrangiana

Minimizar:

$$\sum_{k\in\{1,2\}} \sum_{e\in E} \lambda_e^k z_e \tag{14}$$

Sujeito à restrição 6 e:

$$\sum_{e\in E} z_e \geqslant \sigma \tag{15}$$

Esse problema é facilmente resolvido escolhendo-se entre as arestas $e \in E$, as σ com menor peso $\sum_{k\in\{1,2\}} \lambda_e^k$. Para essas arestas, é dado o valor 1 e para as outras é dado o valor 0.

3.3 Método do Subgradiente

Vamos utilizar o Método do Subgradiente para resolver o dual lagrangiano mostrado na Seção 6. No começo do método, inicializamos λ e o *learning rate* π, então temos um processo iterativo. Em cada iteração recebemos uma solução ótima para (9) usando o λ atual, juntamente com o melhor *upper bound* z_{UB} e o *lower bound* atual $z_{LB}^{(i)}$, e então atualizamos o λ conforme o Algoritmo 1. Se de uma iteração para outra o *lower bound* atual piorar, atualizamos o *learning rate* fazendo $\pi = \pi \cdot \frac{\sqrt{5}-1}{2}$.

Algorithm 1 Subgradient

1: **function** INITIALIZE(λ, π)
2: $\lambda_e^k = 1$, para todo $e \in E$ e $k \in \{1, 2\}$
3: $\pi \leftarrow 2$
4: **return** λ, π
5: **function** UPDATE(x, z, $z_{UB}, z_{LB}^{(i)}$, λ, π)
6: $g_e^k = z_e - x_e^k$ para todo $e \in E$ e $k \in \{1, 2\}$
7: $\alpha = \pi \frac{(z_{UB} - z_{LB})}{\sum_{k \in \{1,2\}} \sum_{e \in E} (g_e^k)^2}$
8: $\lambda_e^k = max(0, \lambda_e^k + \alpha \cdot g_e^k)$, para todo $e \in E$ e $k \in \{1, 2\}$
9: **return** λ

3.4 Heurística Lagrangiana

Na Heurística Lagrangiana recebemos dois ciclos hamiltonianos *tour*1 e *tour*2, além de uma constante σ, então devemos retornar dois ciclos hamiltonianos $newTour1$ e $newTour2$ tais que os mesmos compartilham no mínimo *sigma* arestas. Nosso algoritmo está a seguir:

4 Algoritmo de Solução do Problema

→ Faltou uma breve descrição textual da heurística.

5 Experimento Computacional

5.1 Configuração da Máquina

O problema foi executado ~~num~~ em um 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 gcc (GCC) 11.2.1 20220127 (Red Hat 11.2.1-9), Concorde [2] e o solver QSOpt [3].

Como linguagem de programação, utilizamos C++ [4].

5.2 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 colunas 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.

1: **function** HEURISTIC($tour1, tour2, \sigma$)
2: $equalEdges \leftarrow$ arestas em comum entre $tour1$ e $tour2$
3: $remainingEdges \leftarrow$ arestas que não estão em ao menos um dos tours
4: Ordene as arestas dos conjuntos $equalsEdges$ e $remainingEdges$ por seus custos, onde o custo de uma aresta é a soma dos valores do seu custo em cada tour
5: $edges \leftarrow$ concatenação de $equalsEdges$ e $remainingEdges$, nesta ordem
6: $newTour1 \leftarrow \emptyset$
7: $newTour2 \leftarrow \emptyset$
8: $numSimilar \leftarrow 0$
9: **for** cada aresta e em $edges$ **do**
10: **if** $numSimilar \geq \sigma$ **then**
11: **break**
12: **if** e não forma um subtour em $newTour1$ e $newTour2$ **then**
13: $newTour1 \leftarrow newTour1 \cup \{e\}$
14: $newTour2 \leftarrow newTour2 \cup \{e\}$
15: $numSimilar \leftarrow numSimilar + 1$
16: $edgesTour1 \leftarrow$ arestas do $tour1$ ordenadas por seu custo no $tour1$
17: **for** cada aresta e em $edgesTour1$ **do**
18: se e não formar subtour em $newTour1$, adicione e no mesmo
19: $edges1 \leftarrow$ todas as arestas ordenadas por seu custo no $tour1$
20: **for** cada aresta e em $edges1$ **do**
21: se e não formar subtour em $newTour1$, adicione e no mesmo
22:
23: $edgesTour2 \leftarrow$ arestas do $tour2$ ordenadas por seu custo $tour2$
24: **for** cada aresta e em $edgesTour2$ **do**
25: se e não formar subtour em $newTour2$, adicione e no mesmo
26: $edges2 \leftarrow$ todas as arestas ordenadas por seu custo no $tour2$
27: **for** cada aresta e em $edges2$ **do**
28: se e não formar subtour em $newTour2$, adicione e no mesmo
29: **return** $newTour1$ e $newTour2$

Gostei muito!

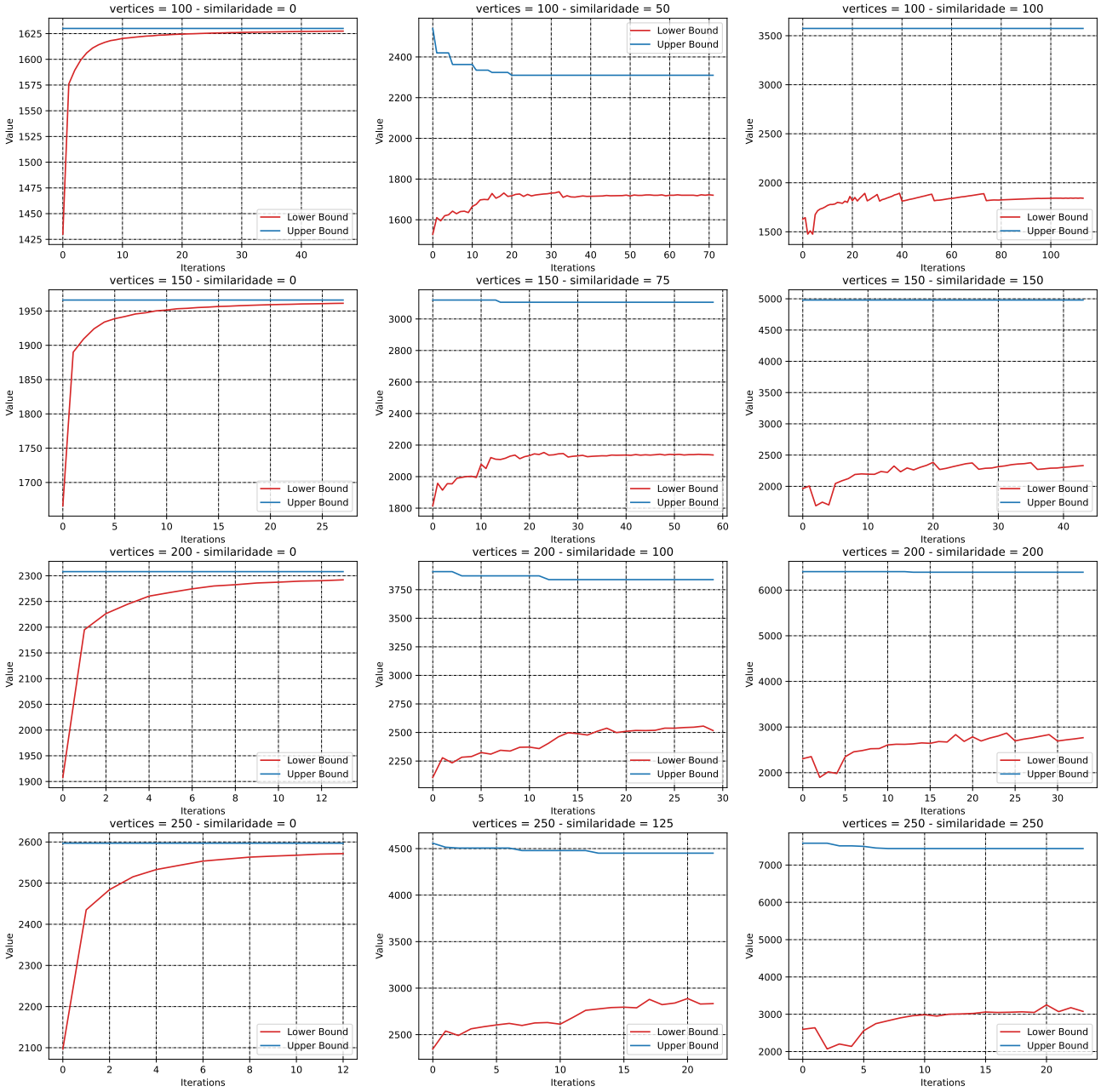
5.3 Geração das Instâncias

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

6 Resultados dos Experimentos

$\|V\|$	σ	z_{UB}	z_{LB}	opt gap	τ
100	0	1630	1627.41	0.002	609
100	100	3574	1842.37	0.940	329
100	50	2310	1720.67	0.343	240
150	0	1966	1961.29	0.002	605
150	150	4980	2331.49	1.136	618
150	75	3106	2137.31	0.453	507
200	0	2308	2292.01	0.007	619
200	100	3838	2516.89	0.525	617
200	200	6392	2766.68	1.310	635
250	0	2597	2571.7	0.010	733
250	125	4451	2833.34	0.571	700
250	250	7442	3076.08	1.419	616

Tabela 1: Resultados dos experimentos computacionais. τé o tempo de processamento em segundos.


vertices = 100 - similaridade = 0
vertices = 100 - similaridade = 50
vertices = 100 - similaridade = 100
vertices = 150 - similaridade = 0
vertices = 150 - similaridade = 75
vertices = 150 - similaridade = 150
vertices = 200 - similaridade = 0
vertices = 200 - similaridade = 100
vertices = 200 - similaridade = 200
vertices = 250 - similaridade = 0
vertices = 250 - similaridade = 125
vertices = 250 - similaridade = 250
Lower Bound
Upper Bound
Value
Iterations

function KSTSP($custos1, custos2, \sigma, \tau_{max}, \pi_{min}, \Delta_{min}$)
  $\lambda_0 \leftarrow 1$
  $\tau \leftarrow 0$
  $\pi \leftarrow 2$
  $\Delta z_{LB} \leftarrow +\infty$
  $\Delta z_{UB} \leftarrow +\infty$
  while $\tau < \tau_{max}$ or $\pi > \pi_{min}$ or ($\Delta z_{LB} > \Delta_{min}$ and $\Delta z_{UB} > \Delta_{min}$) **do**
    $tsp1 \leftarrow$ Solve-TSP($custos1$)
    $tsp2 \leftarrow$ Solve-TSP($custos2$)
    $z \leftarrow$ Resolve-z(λ, σ
    $ciclo1 \leftarrow$ Heuristic($tsp1, tsp2, \sigma$)
    $ciclo2 \leftarrow$ Heuristic($tsp2, tsp1, \sigma$)
    $z_{LB}, \Delta z_{LB} \leftarrow$ Compute-Lower-Bound($ciclo1, ciclo2, z, \lambda$)
    $z_{UB}, \Delta z_{UB} \leftarrow$ Compute-Upper-Bound($ciclo1, ciclo2, z$)
    $\lambda, \leftarrow$ Update($ciclo1, ciclo2, z, .z_{UB}, \lambda, \pi$)
    if $\Delta z_{LB} < 0$ **then**
      $\pi \leftarrow \frac{\sqrt{5}-1}{2}\pi$

7 Análise dos Resultados

Nota-se que um dos passos do Algoritmo 4 é encontrar duas soluções independentes de TSP, um para cada conjunto de distâncias fornecidas. Mas quando o fator de similaridade σ é zero, essa é justamente a solução ótima. Portanto, para tais casos, o algoritmo encontra a solução ótima no primeiro passo. Isso significa que o Upper Bound z_{UB} nunca será atualizado. Já o Lower Bound z_{LB} depende dos multiplicadores λ escolhidos. A solução ótima do Dual Lagrangiano 9 é então atingida quando $\lambda = 0$. No caso em questão, $\sigma = 0$ e portanto a solução do problema da Subseção 3.2.2 é $z = 0$. Isso faz com que o subgradiente g no Algoritmo 3.3 seja sempre negativo, i.e. a solução de fato direciona-se para $\lambda = 0$. Tudo isso pode ser visto nos gráficos de "similaridade = 0". Neles, o Upper Bound z_{UB} é constante, enquanto que o Lower Bound z_{LB} dirige-se para a solução ótima.

Para a solução de todas as instâncias, foi dado um tempo limite de 10 minutos (600 segundos). Em muitos casos, foi tal critério que fez a execução do algoritmo parar. Em alguns casos, o tempo excedeu o limite devido a detalhes de implementação da verificação de limite de tempo. O algoritmo checa tal limite apenas no reinício do loop, de forma que sua nunca é interrompida. Isso poderia ser melhorado, mas para os objetivos da atividade não houve tal necessidade.

Nos outros casos, foi o critério de passo mínimo π que definiu a parada do algoritmo. Toda vez que o Lower Bound z_{LB} piora de uma iteração para a outra, o passo é reduzido. Assim, casos que apresentam bastante oscilação do Lower Bound têm maiores chances de parar por tal critério. Isso é justamente o que observamos nas instâncias de 100 vértices com similaridade σ igual a 50 e 100, e na instância de 150 vértices com similaridade 75.

Em todos os casos, observa-se baixa variação do Upper Bound com o número de iterações. Os casos de similaridade $\sigma = 0$ já foram discutidos acima. Para os outros casos, isso pode indicar que a heurística de factibilização da solução do Dual Lagrangiano não é capaz de gerar instâncias factíveis boas. Também pode indicar que os parâmetros utilizados no algoritmo são inadequados, muito embora sejam necessárias mais investigações e experimentações para ter alguma conclusão sobre a real causa desse comportamento.

Por fim, observa-se que é obtido um gap de otimalidade, apresentado na Tabela 6. Avaliações

quando à qualidade de tais gaps de otimalidade seriam interessantes de um ponto de vista prático, muito embora tal análise tenha que levar em conta também outros fatores como tempo de execução. Portanto, apesar do valor prático, está fora do escopo dessa atividade qualificar tais resultados.

Referências

[1] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, vol. 6. Athena Scientific Belmont, MA, 1997.

[2] W. Cook, "Concorde tsp solver," 2022.

[3] D. Applegate, W. Cook, S. Dash, and M. Mevenkamp, "Qsopt linear programming solver," 2022.

[4] B. Stroustrup, *The C++ programming language*. Pearson Education, 2013.