Submetido para MO824

Atividade 3 - Grupo 8 - MO824

André Soranzzo Mota - RA 166404
Victor Ferreira Ferrari - RA 187890
Gabriel Oliveira dos Santos - RA 197460

Resumo. Este documento apresenta uma relaxação lagrangiana para o 2-TSP, que visa encontrar dois ciclos hamiltonianos com conjuntos distintos de arestas e cuja soma de peso dessas arestas seja mínima. O modelo matemático é apresentado, assim como o processo de obtenção do custo lagrangiano, o método de subgradientes utilizado para obtenção de limitantes duais, e a heurística lagrangiana criada para transformar o dual em uma solução viável para o 2-TSP. Foi realizado a geração de instâncias através de um *script* em *Python* e do software Gurobi. A relaxação lagrangiana forneceu melhores limitantes, porém estourou mais facilmente o tempo limite de execução, possivelmente devido ao tempo exponencial da heurística. O modelo linear inteiro retornou o resultado ótimo, mas aumentou muito tempo de execução conforme aumentamos o tamanho da instância.

Palavras-chave. Relaxação Lagrangiana, Otimização, 2-TSP, Heurística.

1. Introdução

Este trabalho consiste na apresentação de uma relaxação lagrangiana para o problema proposto na Atividade 3 de MO824 2S-2020, e da realização de testes em comparação com a resolução direta do modelo matemático original para o problema.

Chamado 2-TSP, o problema é uma variação do tradicional TSP (*travelling salesman problem*) e busca encontrar dois ciclos hamiltonianos com conjuntos de arestas disjuntos e cuja soma de peso dessas arestas seja mínima.

Para testar o modelo relaxado, foi utilizado o método de subgradientes, baseado no material apresentado em aula e pelo artigo de John E. Beasley [1]. As técnicas retiradas do artigo serão devidamente apontadas.

2. Modelo Matemático

A seguinte variável foi usada no modelo:

x_e^k: Variável de decisão binária associada à presença ($x_e = 1$) ou não ($x_e = 0$) da aresta e no ciclo k, para $k \in \{1, 2\}$ e $e \in E$.

O modelo original do 2-TSP pode ser visto a seguir.

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

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.

Dualizando o conjunto de restrições (4) e introduzindo os multiplicadores de Lagrange para cada restrição ($\forall e \in E$), temos uma relaxação para o 2-TSP, com função objetivo:

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

que podemos simplificar para:

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

Assim, o LLBP para o 2-TSP é o seguinte:

$$\begin{aligned}
\min \quad & \sum_{k\in\{1,2\}}\sum_{e\in E} C_e x_e^k - \sum_{e\in E} \lambda_e && && (1)\\
\text{s.a} \quad & \sum_{e\in\delta(i)} x_e^k = 2 && \forall i \in V, \forall k \in \{1,2\} && (2)\\
& \sum_{e\in E(S)} x_e^k \leq |S| - 1 && \forall S \subset V, \forall k \in \{1,2\} && (3)\\
& x_e^k \in \{0,1\} && \forall e \in E, \forall k \in \{1,2\} && (5)
\end{aligned}$$

onde C_e é o custo lagrangiano do problema, dado por $c_e + \lambda_e \ \forall e \in E$.

O método de subgradientes consiste em otimizar os multiplicadores de Lagrange para encontrar o melhor limitante dual (ou seja, resolver o problema dual lagrangiano) ao deslocar na direção do subgradiente da função objetivo em relação a λ. Uma componente do subgradiente é:

$$g_e = \sum_{k\in\{1,2\}} (x_e^k - 1) \qquad \text{(GRAD)}$$

3. Implementação

A implementação da solução foi feita em *Python*, por meio da construção de um "framework". Uma função genérica que realiza a execução do método de subgradientes recebe um objeto de tipo "Problem", classe abstrata que fornece uma série de métodos que devem ser implementados em classes filhas. O método segue o pseudocódigo visto em aula, com adições propostas em [1].

3.1. Método de Subgradientes

A função objetivo do LLBP consiste da subtração entre dois termos. Como um desses termos é linear, não dependendo das variáveis do problema, não tem impacto na resolução do problema. Assim, ele pode ser adicionado apenas no resultado final. Uma descrição em alto nível do método utilizado pode ser visto no Algoritmo 1.

Algorithm 1: Método de Subgradientes

1: Inicialize valores de controle (π, t_0) e multiplicadores $\lambda_e \ \forall e \in E$.
2: Calcule os custos lagrangianos com o conjunto atual de multiplicadores $\lambda^{(k)}$.
3: Resolva o 2-TSP sem a restrição que garante que os ciclos sejam aresta-disjuntos, obtendo uma solução $x^{(k)}$.
4: Subtraia a soma dos multiplicadores do conjunto atual do custo da solução, para obter o limitante dual $Z_L^{(k)}$ para o 2-TSP.
5: Utilize a heurística lagrangiana vista em Algorithm 2 para obter um limitante primal $Z_U^{(k)}$.
6: Calcule o conjunto de subgradientes $g_e^{(k)}$ segundo a equação (GRAD), a partir de $x^{(k)}$.
7: Calcule o passo, dado por $\alpha^{(k)} = \pi \frac{(1+\varepsilon)Z_U^{(k)} - Z_L^{(k)}}{(\sum_{e \in E} g_e^{(k)})^2}$
8: Atualize o conjunto de multiplicadores: $\lambda_e^{(k+1)} = \max(0, \lambda_e^{(k)} + \alpha^{(k)} g_e^{(k)})$.
9: Retorne ao passo 2 até que um dos critérios de parada seja atendido.
10: Devolva os melhores limitantes dual e primal, com a melhor solução encontrada.

Entre os passos 4 e 5, o valor de π é reduzido pela metade a cada m iterações, iniciando com um valor π_0. Esses valores foram fixados em $\pi_0 = 2, m = 30$ [1]. No passo 7, um fator $1 + \varepsilon$ multiplica o limitante primal para evitar que α fique muito pequeno [1]. ε é fixado em 0.0005. Os multiplicadores são inicializados com zero, valor escolhido arbitrariamente.

Os critérios de parada utilizados incluem tempo limite, otimalidade da solução encontrada, valor mínimo de π (limite de iterações), e **convergência**. Esse último critério ocorre quando todos os componentes do subgradiente são iguais a zero, ou seja, os multiplicadores não sofrerão alteração. Nesse caso, se a solução dual for viável para o problema original, é a solução ótima do problema [1]. Ligado a isso, se em uma determinada iteração um multiplicador $\lambda_e^{(k)}$ tem valor zero e a componente correspondente do subgradiente for negativa, pelo passo 8 o multiplicador não sofrerá alteração. Por isso, pode-se igualar aquela componente a zero [1].

3.2. Heurística Lagrangiana

Implementamos nossa heurística baseada na modelagem do TSP no Gurobi. Dados dois ciclos hamiltonianos c_1 e c_2, escolhemos o ciclo de menor custo, e desconsideramos o outro. Após isso, removemos do grafo as arestas que fazem parte do ciclo que foi mantido, e executamos o algoritmo que resolve o TSP usando o Gurobi para o grafo resultante. Dessa forma, ao término da execução do algoritmo temos dois ciclos hamiltonianos com conjuntos disjuntos de arestas, sendo, portanto, uma solução viável para o 2-TSP. Abaixo é apresentado o pseudo-código, Algoritmo 2.

Algorithm 2: Heurística Lagrangiana

Require: grafo não direcionado G = (V, E) e ciclos hamiltonianos c_1 e c_2
ciclo $\leftarrow \emptyset$
if custo(c_1) $<$ custo(c_2) **then**
  ciclo $\leftarrow c_1$
else
  ciclo $\leftarrow c_2$
end if
for all $(u, v) \in$ ciclo **do**
  remova (u, v) de E
end for
$ciclo_{novo} \leftarrow$ TSP(G)
return ciclo, $ciclo_{novo}$

4. Metodologia de Avaliação

4.1. 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 arquivo `generator.py`, e todas as instâncias foram executadas com a mesma *seed*.

4.2. 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	AMD Ryzen 5 2500U (4C/8T)
Frequência do Clock da CPU	2.00 GHz
RAM	12 GB/2400 MHz

O sistema operacional utilizado foi o Ubuntu 20.04.1 LTS. Foi utilizado como software de execução o *solver* Gurobi Optimizer V9.0.3. Os modelos foram executados com limite de 1800 segundos (30 minutos) e sem limite de memória.

5. 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

Instância	PLI Limite Inferior	PLI Limite Superior	PLI Tempo de Execução (s)	Relaxação Lagrangiana Limite Inferior	Relaxação Lagrangiana Limite Superior	Relaxação Lagrangiana Tempo de Execução (s)
100	19,1645	19,1660	112,9264	18,2633	19,3054	205,8787
150	22,8620	22,8620	471,1312	21,7847	23,0409	1800,2839
200	25,9125	25,9143	1404,9435	23,1658	26,2486	1800,6086
250	28,4179	36,7105	1800,0144	24,4823	29,0124	1802,1623
300	31,6393	inf	1800,1609	26,1999	32,2801	1800,9896

A partir da Tabela 2 podemos observar que para instâncias menores dos problema (200 vértices ou menos) o algoritmo de Programação Linear Inteira (PLI) encontra facilmente a solução ótima em menos de 30 minutos. Entretanto, para instâncias maiores o algoritmo estoura o tempo limite de execução, de 30 minutos, e tem uma parada forçada retornando limitantes inferior e superior não tão bons. Nesse cenário o método de relaxação lagrangiana se apresenta uma abordagem melhor, note que mesmo estourando o tempo limite, são retornados limitantes melhores que o do modelo de PLI. Especificamente para a instância com 300 vértices, o modelo de PLI sequer conseguiu encontrar algum limitante superior, enquanto a relaxação lagrangiana retornou os limitantes 26.1999 e 32.2801.

De modo geral, pode-se dizer que a abordagem usando o método de subgradientes estoura mais facilmente o tempo limite, mas isso se deve ao fato de nossa heurística também ter tempo exponencial. Assim, seria interessante tentar encontrar alguma heurística em tempo polinomial para tentar melhorar esse tempo de execução. Todavia, ainda com essa heurística que executa em tempo exponencial, conseguimos ter bons resultados para instâncias grandes (250 vértices ou mais).

Observando os valores dos limitantes duais e primais encontrados ao longo do tempo, é perceptível a convergência ao ótimo, embora o método não tenha encontrado esse valor em nenhuma instância. Nos testes feitos, apenas um conjunto de hiperparâmetros, descritos na seção 3, foram usados. Modificando esses parâmetros, como o valor inicial dos multiplicadores, o valor de ε ou o número de iterações até a divisão de π, é possível que resultados melhores sejam encontrados dentro do tempo limite.

Referências

[1] J. E. Beasley, "Lagrangean relaxation," in *Modern Heuristic Techniques for Combinatorial Problems* (C. R. Reeves, ed.), ch. 6, pp. 243–303, 605 Third Ave. New York, NY, United States: John Wiley & Sons, 1993.