PLI - Relatório 1: Formulação compacta

Guilherme Guidotti Brandt (235970)

1 Formulação

Considerando a versão direcionada do grafo (adicionando um arco em cada sentido para cada aresta no grafo original), utilizei a seguinte formulação do problema:

$$
\begin{aligned}
(\mathcal{P}) \quad & \min \sum_{(u,v)\in A} \gamma c_{uv} x_{uv} + \sum_{(u,v)\in A} c_{uv} y_{uv} \\
& \sum_{v\in V} r_v \geq 3 && (1)\\
& \sum_{v\in\delta^-(u)} y_{vu} = 1 - r_u && \forall u \in V \quad (2)\\
& \sum_{v\in\delta^+(u)} d_v y_{uv} \leq r_u(B - d_u) && \forall u \in V \quad (3)\\
& \sum_{v\in\delta^+(u)} x_{uv} = r_u && \forall u \in V \quad (4)\\
& \sum_{v\in\delta^-(u)} x_{vu} = r_u && \forall u \in V \quad (5)\\
& t_u - t_v + |V|(x_{uv} - y_{vv_0}) \leq |V| - 1 && \forall (u,v) \in A, v \neq v_0 \quad (6)\\
& x_{uv} \in \{0,1\} && \forall (u,v) \in A\\
& y_{uv} \in \{0,1\} && \forall (u,v) \in A\\
& r_v \in \{0,1\} && \forall v \in V\\
& t_v \in \mathbb{R}_{\geq 0} && \forall v \in V
\end{aligned}
$$

Onde A é o conjunto dos arcos do grafo orientado, V é o conjunto de vértices no grafo, e as constantes $c_{uv} = c_{vu}$ na função de custo correspondem ao custo c_e da aresta e que conecta u e v no grafo original.

1.1 Variáveis

As variáveis são:

- x_{uv}, indica se o arco (u, v) está sendo usado no ciclo;

- y_{uv}, indica se o arco (u, v) está sendo usado em alguma estrela;
- r_v, indica que o vértice v está no ciclo;
- t_v, indica a ordem do vértice v no ciclo.

1.2 Restrições

Podemos dividir as restrições da formulação em três grupos:

- Restrições de partição, que garantem que a partição dos vértices respeita as condições do problema;
- Restrições de topologia, que garantem que a topologia da solução atende os critérios gerais do problema (exceto pela existência de subciclos);
- Uma restrição de eliminação de subciclo, que completa as restrições de topologia, garantindo que a solução contém apenas um ciclo.

1.2.1 Restrições de partição

A restrição 1 garante que a partição dos vértices tem pelo menos três partes (i.e. o ciclo tem tamanho maior ou igual a 3).

A restrição 3 garante que a soma dos pesos dos vértices de uma estrela não ultrapasse o valor limite B.

1.2.2 Restrições de topologia da solução

A restrição 2 tem função dupla, garantindo que exatamente um arco de estrela entre em cada vértice fora do ciclo, e também que nenhum arco de estrela entra em um vértice do ciclo.

A restrição 3 também garante que nenhuma aresta de estrela sai de um vértice fora do ciclo.

As restrições 4 e 5 garantem que todo vértice no ciclo tem grau de entrada e saída (ignorando as arestas de estrela) iguais a 1, e analogamente que todo vértice fora do ciclo tem grau zero (i.e. um arco do ciclo nunca incide sobre um vértice fora do ciclo).

1.2.3 Restrição de eliminação de subciclos

A restrição 6 é análoga à de eliminação de subciclos na formulação Miller-Tucker-Zemlin do problema do caixeiro viajante (TSP), e impõe uma ordem aos vértices de forma que qualquer ciclo no grafo necessariamente inclua um vértice particular (que é fixo para um dado grafo). Como os vértices no ciclo têm grau de entrada e saída iguais a 1 (pelas restrições de topologia), isso é suficiente para garantir que a solução contém exatamente um ciclo.

Na formulação do TSP, como o ciclo é hamiltoniano, basta escolher um vértice qualquer. No nosso caso, como nem todo vértice está no ciclo, precisamos de alguma forma de escolher um vértice que esteja no ciclo.

Para isso, escolhemos um vértice v_0 qualquer (que não necessariamente está ciclo), e como vértice que deve estar no ciclo usamos v_0 ou o vértice do ciclo que atende v_0 (que é único). Podemos identificar o vértice que atende v_0 através dos arcos de estrela.

Dessa forma, a restrição 6 força que todo vértice no ciclo tenha ordem maior que seu predecessor (i.e. a cauda do arco que entra no vértice), como única exceção o vértice v_0 ou o vértice que atende v_0 (i.e. o vértice v tal que $y_{vv_0} = 1$). Isso é suficiente para garantir que existe um único ciclo na solução.

Como uma melhoria, podemos ainda fortalecer a restrição 6 da seguinte forma [Desrochers e Laporte 1991]:

$$t_u - t_v + |V|(x_{uv} - y_{vv_0}) + (|V| - 2)x_{vu} \leq |V| - 1 \qquad \forall (u, v) \in A, v \neq v_0$$

2 Heurística

Para definir um cut-off inicial razoável, implementei uma heurística gulosa para o problema. A heurística consiste essencialmente em tentar encontrar estrelas no grafo que minimizem o custo da solução de forma gulosa (ou seja, a cada passo adicionando a melhor estrela encontrada).

Para encontrar uma estrela com custo baixo, a heurística escolhe, dado um centro, as arestas com a menor razão custo/peso do vértice. Uma melhoria possível (não cheguei a implementar) seria resolver um problema da mochila para encontrar o melhor empacotamento de vértices na estrela para cada centro.

A cada passo, a heurística adiciona a estrela à solução, adicionando também uma aresta entre o centro da última estrela adicionada e o centro da atual.

O algoritmo 1 descreve o procedimento da heurística.

Algoritmo 1: Heurística gulosa para o problema

Data: G, um grafo; $c : E(G) \to \mathbb{Z}$, função custo das arestas de G;
$d : V(G) \to \mathbb{Z}$, função de peso dos vértices de G
Result: O conjunto de arestas da solução
$S \leftarrow V(G)$;
$R \leftarrow \emptyset$;
$c_0 \leftarrow \emptyset$;
$c \leftarrow \emptyset$;
while $S \neq \emptyset$ **do**
  $H \leftarrow$A estrela em $G[S]$ com menor custo (considerando o custo da aresta de ciclo a ser adicionada), com centro v;
  $S \leftarrow S - V(H)$;
  $R \leftarrow R \cup E(H)$;
  if $c \neq \emptyset$ **then**
    $R \leftarrow R \cup \{(c, v)\}$;
  else
    $c_0 \leftarrow v$;
  end
  $c \leftarrow v$;
end
return R

3 Experimentos computacionais

Executei o programa em uma máquina com o sistema operacional Manjaro Linux, com 32 GB de RAM e um processador Intel i9-12900K (especificado na tabela 1), utilizando a build v10.0.3rc0 (linux64) do Gurobi, com a licença acadêmica.

Modelo	i9-12900K
Fabricante	Intel®
Microarquitetura	Alder Lake
Núcleos	8 performance / 8 eficiência
Threads	24
Frequência	3.20GHz (performance) / 2.40 GHz (eficiência)
Cache	30 MB Intel® Smart Cache
Cache L2	14 MB

Tabela 1: Especificação da CPU

Utilizei as instâncias disponibilizadas junto ao código de exemplo, exceto pelas duas últimas instâncias de 100 nós, visto que as três instâncias de 100 nós que foram executadas atingiram o tempo limite e foram suficientes para analisar o desempenho do modelo.

A tabela 2 mostra os resultados obtidos para cada instância, identificada

pelo número de nós (coluna N) e o número da instância dentro do grupo de nós (coluna Inst.). A linha N = 10, Inst. = 1 corresponde à instância `mo420_network_design_1_10_100_20_80_1`, a linha N = 20, Inst. = 3 corresponde à instância `mo420_network_design_3_20_100_20_80_2`, etc.

Os tempos foram medidos utilizando a classe `std::chrono::steady_clock` da biblioteca `<chrono>` da biblioteca padrão do C++ 11, com precisão de milissegundos.

N	Inst.	Branch-And-Bound			Heurística		Tempo
		Valor	Lim.	Nós	Valor	Tempo	
10	1	23432	–	132	30613	0ms	159ms
	2	29460	–	1	40066	0ms	9ms
	3	29315	–	1	44758	0ms	23ms
	4	32767	–	1	36398	0ms	41ms
	5	30888	–	1	34736	0ms	17ms
20	1	68938	–	571	109846	0ms	431ms
	2	83628	–	6005	120438	0ms	949ms
	3	74927	–	1537	114112	0ms	339ms
	4	73473	–	8362	98290	0ms	633ms
	5	73429	–	1112	119636	0ms	262ms
50	1	244058	–	408816	336160	4ms	163,371s
	2	257081	–	43240	367513	1ms	24,106s
	3	258745	249187	8570011	390803	1ms	**3600s**
	4	250595	–	155381	396860	1ms	66,510s
	5	238698	–	14754	389217	1ms	4,609s
100	1	570825	529124	587894	1081590	20ms	**3600s**
	2	561811	537250	1238796	1167600	9ms	**3600s**
	3	514868	432015	755331	822091	12ms	**3600s**

Tabela 2: Resultados computacionais

De forma geral, o programa é consistentemente rápido (tempo < 1s) para resolver instâncias de até 20 nós (grupo N = 20). O resultado é variado para 50 nós, com casos extremos na instância 5, que demorou cerca de 5 segundos, e na instância 3, que atingiu o tempo limite; em geral, as instâncias de 50 nós demoram menos que três minutos para convergir.

A instância de 50 nós que atingiu o tempo limite terminou o processo com um gap de 3,69%. Analisando os logs gerados, o modelo aparenta ter tido dificuldade para aumentar o valor do limite inferior.

Todas as instâncias de 100 nós atingiram o tempo limite, com gaps de 7.3055%, 4.3719% e 16.0922% (respectivamente, para as instâncias 1, 2 e 3).

Referências

[Desrochers e Laporte 1991]DESROCHERS, M.; LAPORTE, G. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, v. 10, n. 1, p. 27–36, 1991. ISSN 0167-6377. Disponível em: <https://www.sciencedirect.com/science/article/pii/0167637791900832>.