PLI - Relatório 2: Formulação por cortes

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:

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

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.

A formulação é, na maior parte, idêntica à formulação compacta apresentada na primeira parte do projeto. A única diferença entre as duas é que esta formulação não tem as variáveis t_u, e a restrição de eliminação de subciclos de Miller-Tucker-Zemlin é substituída pela restrição 6, de conexidade, análoga à descrita por Dantzig, Fulkerson e Johnson para o problema do caixeiro viajante.

1.1 Variáveis

As variáveis são idênticas às da formulação compacta, exceto que, nessa formulação, não temos variáveis t_u indicando a ordem dos nós no ciclo:

- 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 se o vértice v está no ciclo;

1.2 Restrições

Assim como a formulação compacta, podemos dividir as restrições dessa 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 conexidade, que completa as restrições de topologia, garantindo que a solução é conexa (e por consequência que não existem subciclos).

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 conexidade

A restrição 6 é análoga à restrição de Dantzig-Fulkerson-Johnson para o problema do caixeiro viajante, em sua forma de cortes de aresta para grafos orientados. No nosso caso, a conexidade do grafo envolve duas variáveis: x e y, correspondentes aos arcos do ciclo e das estrelas. Em particular, para todo subconjunto S dos vértices do grafo, pelo menos um arco aresta de circuito sai de S, ou pelo menos um arco de estrela entra ou sai de S [1].

A separação para essa restrição é análoga à da restrição original, e pode ser implementada de forma (relativamente) eficiente com árvore de Gomory-Hu.

2 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

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.

A tabela 3 mostra os resultados obtidos comparados aos resultados da formulação compacta utilizada na parte 1.

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

O programa se sai bem com instâncias de até 20 nós, e apresenta tempo comparável à formulação compacta, com uma única exceção notável na instância N = 20, Inst. = 4, onde a formulação compacta dá o resultado ótimo em 633ms, enquanto a formulação por cortes leva 5,489s (quase 9 vezes mais lento).

[1] Mas não é possível determinar que algum dos casos dos arcos de estrela se aplique a todo corte; por exemplo, num corte que contenha apenas nós do ciclo, nenhum arco de estrela entra, e para um corte apenas com nós de estrela, nenhum arco de estrela sai.

N	Inst.	Branch-And-Cut			Heurística		Tempo
		Valor	Lim.	Nós	Valor	Tempo	
10	1	23432	–	1	30613	0ms	41ms
	2	29460	–	1	40066	0ms	6ms
	3	29315	–	1	44758	0ms	3ms
	4	32767	–	1	36398	0ms	19ms
	5	30888	–	1	34736	0ms	5ms
20	1	68938	–	223	109846	0ms	219ms
	2	83628	–	8195	120438	0ms	1483ms
	3	74927	–	5109	114112	0ms	837ms
	4	73473	–	19391	98290	0ms	5489ms
	5	73429	–	239	119636	0ms	146ms
50	1	247468	220010	1041381	336160	1ms	**3600s**
	2	257630	247785	2468317	367513	1ms	**3600s**

Tabela 2: Resultados computacionais

Por outro lado, para as instâncias com 50 nós ou mais, a formulação por cortes curiosamente apresenta desempenho muito inferior ao da formulação compacta (em contraste ao TSP, onde a formulação por cortes é mais eficiente), atingindo o tempo limite em duas instâncias para as quais a formulação compacta encontra soluções ótimas em menos de três minutos (ou seja, um aumento de mais de 20 vezes no tempo).

N	Inst.	Cortes			Compacta		
		Valor	Lim.	Tempo	Valor	Lim.	Tempo
10	1	23432	–	41ms	23432	–	159ms
	2	29460	–	6ms	29460	–	9ms
	3	29315	–	3ms	29315	–	23ms
	4	32767	–	19ms	32767	–	41ms
	5	30888	–	5ms	30888	–	17ms
20	1	68938	–	219ms	68938	–	431ms
	2	83628	–	1,483s	83628	–	949ms
	3	74927	–	837ms	74927	–	339ms
	4	73473	–	5,489s	73473	–	633ms
	5	73429	–	146ms	73429	–	262ms
50	1	247468	220010	**3600s**	244058	–	163,371s
	2	257630	247785	**3600s**	257081	–	24,106s

Tabela 3: Resultados computacionais comparados à formulação compacta.