O problema de projeto de redes não capacitadas com mono-produto e custos fixos: comparação entre formulações

Rhogger Freitas Silva[1], Jordania Louse Silva Alves[2], Rodrigo Francisco Borges Lourenço[3], Fabíola Medeiros Costa[4], Darlan Marques da Silva[5,6]

[1]Graduando do curso de Engenharia de Software, Universidade de Rio Verde. Aluno de Iniciação Científica – PIBIC.
[2] Prof. Dr(a). do Departamento de Engenharia de Produção, Universidade Federal do Amazonas, Manaus-AM.
[3] Prof. Dr. da Faculdade de Engenharia Mecânica, Universidade de Rio Verde.
[4]Co-orientadora, Prof. Dr(a). da Faculdade de Engenharia Mecânica, Universidade de Rio Verde.
[5]Doutorando em Engenharia de Produção/PPGEP-UFMG, Universidade Federal de Minas Gerais, Belo Horizonte-MG.
[6] Orientador, Prof. MSc. da Faculdade de Engenharia de Produção, Universidade de Rio Verde. darlan@unirv.edu.br.

Reitor:
Prof. Me. Alberto Barella Netto

Pró-Reitor de Pesquisa e Inovação:
Prof. Dr. Carlos César E. de Menezes

Editor Geral:
Prof. Dra. Andrea Sayuri Silveira Dias Terada

Editores de Seção:
Profa. Dra. Ana Paula Fontana
Prof. Dr. Hidelberto Matos Silva
Prof. Dr. Fábio Henrique Baia
Pra. Dra. Muriel Amaral Jacob
Prof. Dr. Matheus de Freitas Souza
Prof. Dr. Warley Augusto Pereira

Fomento:
Programa PIBIC/PIVIC UniRV/CNPq 2022-2023

Resumo: O problema de projeto de redes não capacitadas com mono-produto e custos fixos apresenta importantes aplicações em sistemas logísticos e de telecomunicações. Existem diversas formulações para retratar esta abordagem. Visto isto, surgiu uma lacuna em implementar em Python duas formulações diferentes (M e S) e compará-las em relação a duas variáveis resposta: o GAPs de Otimalidade (%) e o tempo de execução (s), a partir de 10 instâncias diferentes. Com os resultados, foi possível demonstrar que o modelo mais robusto (M) apresentou menores GAPs de Otimalidade (%). Já os resultados do tempo execução (s), mostra evidências para as quais instâncias mais complexas apresentarem um tempo de execução maior para o modelo S. É válido destacar que esta pesquisa está inserida dentro de um projeto maior, sendo um fragmento do projeto global.

Palavras-Chave: Problema logístico de grande escala. Comparação entre formulações. Implementação em Python.

The problem of designing uncapacitated networks with single-product and fixed costs: comparison between formulations

Abstract: *The problem of designing uncapacitated networks with a single product and fixed costs has essential applications in logistics and telecommunications systems. There are several formulations to portray this approach. Given this, a gap arose in implementing two different formulations in Python (M and S) and comparing them about two response variables, the Optimality GAPs (%) and the execution time (s), from 10 instances many different. With the results, it was possible*

to demonstrate that the most robust model (M) presented lower Optimality GAPs (%). The results of execution time (s) show evidence that more complex instances give a longer execution time for model S. It is worth highlighting that this research is part of a larger project, being a fragment of the global project.

Keywords: *Large-scale logistical problem. Comparison between formulations. Implementation in Python.*

Introdução

O problema de projeto de redes não capacitadas com mono-produto e custos fixos consiste em projetar uma rede transporte de forma que os fornecedores sejam capazes de suprir com produtos as demandas no menor custo possível, tanto em questões de transporte quanto de instalação da infraestrutura. Tal problema é também conhecido como *single commodity uncapacitated fixed charge network design problem ou ainda uncapacitated fixed charge network design problem* (*UFCNDP).*

Problemas de projeto de redes de fluxo estão presentes em diferentes contextos práticos. Pode-se encontrar com facilidade em sistemas de transporte, de telecomunicações e de energia. A ideia é estabelecer uma determinada rede de conexões, por exemplo: estradas, fibras ópticas, linhas de distribuição, que permita transportar o fluxo de algum produto ou *commoditie*, como pessoas, pacotes de dados, energia, carga, entre outros, afim de satisfazer alguma demanda específica.

O UFCNDP inicialmente formulado por Hirsch e Dantzig (1968) teve destaque nas primeiras duas décadas seguintes, sendo tratado por trabalhos como Barr *et al.* (1981), Cabot; Erenguc (1984), Magnanti *et al*. (1986), Palekar *et al*. (1990), e Guisewite e Pardalos (1990). Nas últimas duas décadas, os trabalhos de Cruz *et al*. (1998), Ortega e Wolsey (2003), Gendron (2019) e Zetina *et al*. (2019) são as principais referências da literatura.

Vários modelos foram desenvolvidos ao longo dos anos. Sendo assim, existe uma lacuna em realizar a comparação entre modelos UFCNDP e verificar o quão eficaz um é melhor em relação ao outro. Contudo, o objetivo geral deste estudo é realizar a comparação entre dois modelos: M e S (propostos por, respectivamente, Rardin e Parker (1982) e Ortega (2003)).

Material e Métodos

A primeira formulação para o UFCNDP, seja um grafo (rede) direcionado D = (N, A) com os conjuntos de nós N= {1, ..., n} e de arcos A ⊆ { (i, j) ∈ N x N : i ≠ j}. Cada arco (i, j) A possui um custo unitário de transporte $c_{ij} \geq 0$ e um custo fixo $a_{ij} \geq 0$ de instalação; enquanto que cada nó de oferta j ∈ N tem um valor $d_j > 0$, ou como nó de oferta, quando $d_j < 0$, ou como nó de transbordo, quando $d_j = 0$. Para termos um problema viável, temos que $\Sigma_{j \in N} d_j = 0$. Definimos também K+ = { j ∈ N: $d_j > 0$}, K- = { j ∈ N: $d_j < 0$}, e K^0 = { j ∈ N: $d_j = 0$} como os conjuntos de nós de demanda, oferta, e transbordo, respectivamente, sendo ainda D = $\Sigma_{j \in K+} d_j$.

Usando as variáveis de decisão binárias $x_{ij} \in \{0, 1\}$ e de fluxo $f_{ij} \geq 0$ para representar se um arco (i, j) ∈ A é instalado ($x_{ij} = 1$) ou não ($x_{ij} = 0$), e a quantidade de fluxo passando pelo arco (i, j), respectivamente, pode-se formular o UFCNDP como (ORTEGA; WOLSEY, 2003):

$$\emptyset_S(f, x) = min \sum_{(i,j) \in A} \left(c_{ij} f_{ij} + a_{ij} x_{ij}\right) \quad (1)$$

$$s.a.: \sum_{(i,j) \in A} f_{ij} - \sum_{(j.i) \in A} f_{j,i} = d_j \qquad \forall\, j \in N \quad (2)$$

$$f_{ij} \leq D x_{ij} \qquad \forall\, (i,j) \in A \quad (3)$$

$$x_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A \quad (4)$$

$$f_{ij} \geq 0 \qquad \forall\, (i,j) \in A \quad (5)$$

A função objetivo (1) minimiza os custos totais de instalação de rede de fluxo. As restrições (2) são de conservação ou balanço de fluxo pela rede. Se $d_j < 0$, então d_j unidades de fluxo sairão do nó $j \in N$; enquanto que se $d_j > 0$, então d_j unidades de fluxo serão deixados no nó $j \in N$; sendo que se $d_j = 0$, então a mesma quantidade de fluxo que entra no nó $j \in N$ sairá dele. As restrições (3) são restrições de ativação, exemplo, só pode existir fluxo passando no arco $(i, j) \in A$, se o arco (i, j) estiver instalado. Finalmente, restrições (4) e (5) mostram o domínio das variáveis.

Para a segunda formulação indexamos as variáveis de fluxo f_{ij} pelo nó de demanda ou *commodity* ao qual o fluxo está destinado, dando origem às variáveis de decisão $g^k_{ij} \geq 0$, $(i, j) \in A$, e $k \in K+$, representando a quantidade de fluxo passando pelo arco (i, j) com destino a k. Definimos ainda as variáveis de decisão $q^k_j \geq 0$ representando a quantidade de demanda com destino a $k \in K+$ atendida pelo nó de oferta $j \in K-$. A formulação M proposta por Rardin e Parker (1982) é apresentada.

$$min \sum_{k \in k^+} \sum_{(j.i) \in A} c_{ij}\, g^k_{ij} + \sum_{(j.i) \in A} a_{ij} x_{ij} \qquad (6)$$

$$s.a.: \sum_{(i,j) \in A} g^k_{ij} + q^k_j = \sum_{(j.i) \in A} g^k_{ji} \qquad \forall\, j \in k^-, k \in k^+ \qquad (7)$$

$$\sum_{(i,j) \in A} g^k_{ij} - \sum_{(j.i) \in A} g^k_{ji} = 0 \qquad \forall\, j \in N \backslash k^-, k \in k^+ : j \neq k \qquad (8)$$

$$\sum_{(i,k) \in A} g^k_{ik} - \sum_{(k.j) \in A} g^k_{kj} = d_k \qquad \forall\, k \in k^+ \qquad (9)$$

$$g^k_{ij} \leq d_k\, x_{ij} \qquad \forall\, k \in k^+ \; (i,j) \in A \qquad (10)$$

$$\sum_{j \in k^-} q^k_j = d_k \qquad \forall\, k \in k^+ \qquad (11)$$

$$\sum_{k \in k^+} q^k_j = -d_j \qquad \forall\, k \in k^- \qquad (12)$$

$$g^k_{ij} \geq 0 \qquad \forall\, k \in k^+ \; (i,j) \in A \qquad (13)$$

$$q^k_j \geq 0 \qquad \forall\, j \in k^-, k \in k^+ \qquad (14)$$

$$x_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A \qquad (15)$$

A função objetivo (6) tem o mesmo significado da função objetivo (1), enquanto as restrições (7) - (9) são equivalentes àquelas de conservação de fluxo (2), porém agora referentes ao produto $k \in K+$. As restrições (10) são as de ativação de arcos, ao passo que as restrições (11) e (12) computam a distribuição de demanda do nó $k \in K+$ entre os nós de oferta $j \in K-$ e vice-versa, respectivamente. Finalmente, as demais restrições apresentam o domínio das variáveis de decisão.

Foi gerado as coordenadas dos nós de 10 instâncias de forma aleatória numa área quadrada de 100 x 100 com o número de nós da rede variando de |N| = {10, 20, 30, 40, 50}. Depois, calculada a distância Euclidiana entre todos os pares de nós, porém arredondando a distância para o valor inteiro mais próximo. Para determinar o conjunto de arcos, usou a triangulação de Delaunay que encontra as arestas dos triângulos cujas circunferências não têm nenhum nó de N dentro. Fixou o custo fixo de cada arco proporcionalmente ao seu comprimento multiplicado por um fator igual a 50. O número de nós de demandas selecionados aleatoriamente do conjunto N foi selecionado de acordo com | K+ | = {5, 10, 15}, sendo as demandas d_j, $j \in K+$, geradas seguindo uma distribuição uniforme U[1, 10] de valores inteiros. Os nós fontes do conjunto K- foram selecionados aleatoriamente entre N\ K+ de forma que a demanda total seja aleatoriamente distribuída entre eles, porém com o sinal trocado.

Com a geração das instâncias em extensão .py e a implementação das formulações foram através do Windows 7 Ultimate com Intel(R) Core(TM) i5 − 3337U CPU @1:80 GHz processador, 6

GB RAM, e 64x arquitetura. O software Python 3:8 e os pacotes computacionais como Python-MIP foram aplicados para resolver os modelos. Posteriormente foram comparados para cada instância em cada modelo por meio das variáveis respostas: tempo de processamento (s) e Gap de Otimalidade (%).

Resultados e Discussão

Através da Tabela 1 é possível verificar as dez instâncias (i1, i2, i3, ..., i10) geradas que foram codificas de acordo com cada configuração. Para cada configuração, tem-se na segunda coluna (N) a quantidade de nós, terceira coluna (A) a quantidade de arcos, quarta coluna a quantidade de nós de demanda (K+) e quinta coluna (K-) a quantidade de nós de oferta. As colunas subsequentes já encontram-se os resultados obtidos dos modelos S e M em relação ao GAP-LP (%) e tempos (s).

Devido às limitações computacionais, a quantidade de nós de oferta gerados por configuração foram restringidas quanto ao tempo disponível para compilar os modelos no software e a complexidade das árvores geradas. Para todos os resultados, criou-se gráficos para uma melhor compreensão dos valores obtidos, explorando melhor as variáveis.

A Figura 1 evidencia os GAP de *Linear Programming* (*LP*) em porcentagem. Os resultados sugerem que o modelo M por apresentar uma quantidade maior de variáveis e restrições, chegou-se a valores menores para todas as instâncias com variação de 0% a 27,841%. Por outro lado, o modelo S sustenta GAP-LP (%) entre 68,075% a 81,362%.

Tabela 1 – Resultados gerais obtidos entre a comparação das formulações S e M

Código	N	A	K+	K-	GAP-LP (%)		Tempo (s)	
					S	M	S	M
i1	10	44	5	2	68,075	0,000	0,67	0,06
i2	10	44	5	2	74,177	25,691	0,59	1,19
i3	20	98	10	3	69,226	12,454	2,53	11,88
i4	20	102	10	2	70,672	7,680	4,05	7,51
i5	30	160	10	5	81,362	23,033	255,18	441,29
i6	30	152	10	5	75,982	27,841	39,33	160,92
i7	40	220	15	4	76,837	10,277	24.355,13	730,05
i8	40	218	15	4	74,280	5,905	60,11	56,45
i9	50	270	15	3	71,244	6,278	5.025,92	34,85
i10	50	274	15	3	73,279	12,447	514,07	403,45

* nota: Resultados obtidos por meio das 10 instâncias geradas que apresentam o GAP-LP (%), B&B e Time (s) para ambos os modelos, S e M.
Fonte: autoria própria

Um fator interessante é que a complexidade em resolver o problema não se encontra apenas no tamanho da rede, mas também na quantidade de nós de oferta (Zetina *et al.*, 2019). Pela Figura 1 a uma tendência de instâncias com quantidades de nós moderados (i5 e i6) resultar em elevados GAP-LP (%) para o modelo S (81,362%) e M (27,841%).

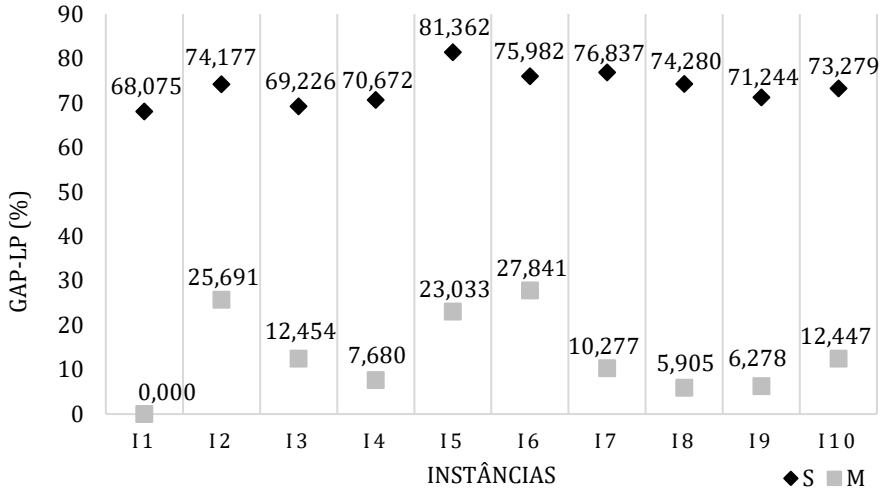
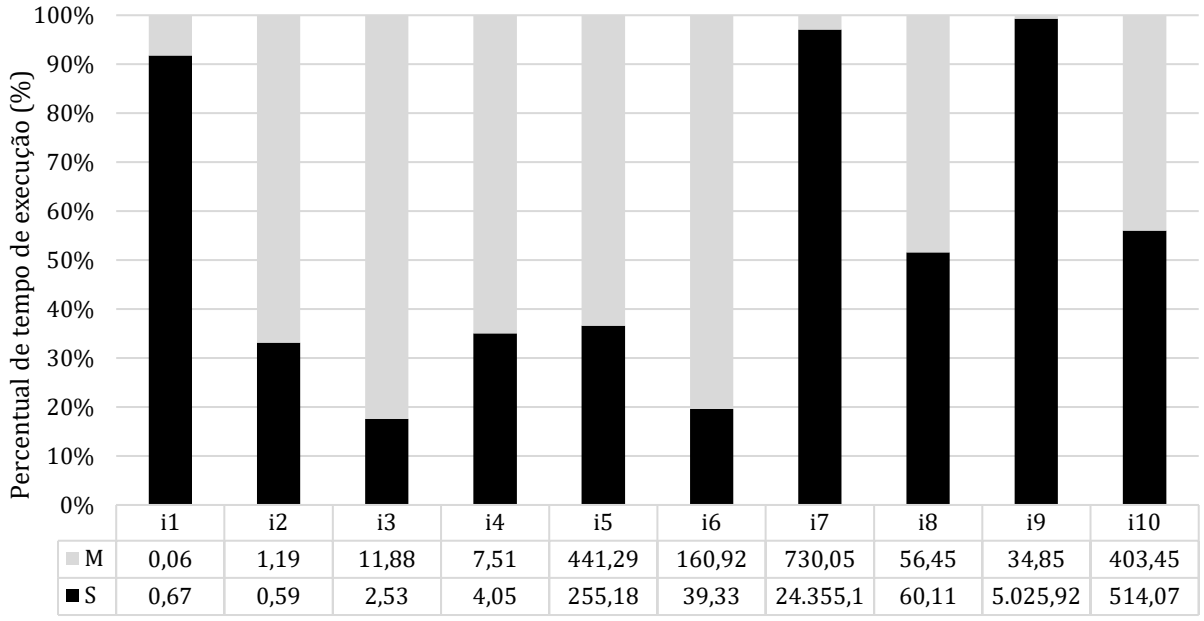


Figura 1 – Apresentação dos GAP-LP (%) por cada instância para os modelos S e M
Fonte: autoria própria

A Figura 2 apresenta os tempos na legenda para cada modelo, S e M, de acordo com as instâncias. É sabido que as redes mais densas sugerem um maior tempo para solucionar o problema em virtude da complexidade (Gendron, 2019). Assim, o gráfico demonstra a proporção de tempo (s) gasto de cada modelo pelas respectivas instâncias. Destaca-se a grande discrepância nas instâncias i1, i7 e i9, as quais os modelos S consomem mais de 90% do tempo total quando comparado ao modelo M. Outro fator, é que a configuração i7 consume 24.355,13 segundos para finalizar o processamento.



	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10
M	0,06	1,19	11,88	7,51	441,29	160,92	730,05	56,45	34,85	403,45
S	0,67	0,59	2,53	4,05	255,18	39,33	24.355,1	60,11	5.025,92	514,07

Figura 2 – Proporção de tempo gasto para solucionar os modelos S e M para cada instância e os valores dos respectivos tempos (s) gastos nas instâncias (abaixo das barras gráficas)
Fonte: autoria própria

Conclusão

Ao longo destas implementações, vimos as dificuldades e facilidades de implementação de cada um dos métodos propostos. Vimos também o desempenho de cada método na resolução do problema proposto. O modelo M, por ser mais robusto consegue chegar a valores melhores em relação ao GAP de Otimalidade (%) como já era esperado. Os tempos computacionais destacam uma tendência da formulação M demandar um maior tempo computacional em instâncias menores, exceto para a primeira instância (i1). Contudo, em instâncias maiores a formulação S demonstra ser menos eficiente, gastando um maior tempo para a execução.

Agradecimentos

A pesquisa apresenta como agradecimento ao apoio financeiro da UniRV-PIBIC.

Referências Bibliográficas

BARR, R.; GLOVER, F.; KLINGMAN, D. A new optimization method for large scale fixed charge transportation problems. **Operations Research**, 29:448–463, 1981.

CABOT, A.; ERENGUC, S. Some branch-and-bound procedures for fixed-cost transportation problems. **Naval Research Logistics Quarterly**, 31:145–154, 1984.

CRUZ, F.; SMITH, J.; MATEUS, G. Solving to optimality the uncapacitated fixed-charge network flow problem. **Computers & Operations Research**, 25(1):67–81, 1998.

GENDRON, B. Revisiting lagrangian relaxation for network design. **Discrete Applied Mathematics**, 261:203–218, 2019.

GUISEWITE, G.; PARDALOS, P. Minimum concave-cost network flow problems: applications, complexity, and algorithms. **Annals of Operations Research**, 25:75–100, 1990.

MAGNANTI, T. L.; MIREAULT, P.; WONG, R. T. **Tailoring Benders decomposition for uncapacitated network design**. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 112–154, 1986.

ORTEGA, F; WOLSEY, L. A. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. **Networks**, 41(3):143–158, 2003.

PALEKAR, U. S.; KARWAN, M. H.; ZIONTS, S. A branch-and-bound method for the fixed charge transportation problem. **Management Science**, 36(9):1092–1105, 1990.

HIRSCH, W.; DANTZIG, G. The fixed charge problem. **Naval Research Logistics Quarterly**, 15:413–424, 1968.

ZETINA, C. A.; CONTRERAS, I.; CORDEAU, J. F. Exact algorithms based on ben-ders decomposition for multicommodity uncapacitated fixed-charge network design. **Computers & Operations Research**, 111:311–324, 2019.