AVALIAÇÃO DE DESEMPENHO COMPUTACIONAL DE ALGORITMOS MATEMÁTICOS PARA RESOLUÇÃO DO PROBLEMA DE FLUXO DE CUSTO MÍNIMO

Kalebe Daniel Elisio Scheidt, Moacyr Carlos Possan Jr

INTRODUÇÃO

As redes de infraestrutura desempenham um papel vital na sociedade contemporânea, levantando a questão central de como mover recursos de maneira eficiente e economicamente viável através de fluxos. Neste contexto destaca-se o clássico Problema de Fluxo de Custo Mínimo (MCFP), que busca determinar a distribuição de fluxo de menor custo em um grafo direcionado sujeito a restrições de equilíbrio de massa e limites de capacidade nos arcos. Fundamentado na teoria de Ahuja et al. (1993) e nos estudos experimentais de Kovács (2015), este trabalho investiga o desempenho computacional de algoritmos específicos para a resolução de tal problema. Para garantir eficiência computacional, a metodologia adota um ambiente em Julia integrado via PythonCall a *solvers* especializados escritos em C++, permitindo a manipulação e resolução de instâncias de *benchmark* sobre a estrutura de dados dos grafos.

DESENVOLVIMENTO

A pesquisa foi conduzida utilizando principalmente como base as obras *Network Flows: Theory, Algorithms, and Applications* (Ahuja, 1993) e o artigo *Minimum-cost flow algorithms: an experimental evaluation* (Kovács, 2015). As especificações técnicas da máquina utilizada na pesquisa são as seguintes: CPU Intel Core i5-12450HX, 8GB RAM, tendo como sistema operacional Linux Fedora Workstation 44. Dada a natureza do problema de fluxo de custo mínimo, os algoritmos utilizados para a avaliação de *benchmark* foram o NS – *Network Simplex*; *SCC - Simple Cycle Cancelling* e *CAS - Capacity-Scaling Algorithm*.

O ambiente de desenvolvimento escolhido foi o VSCodium com linguagem de programação Julia em paralelo com o pacote PythonCall para importação direta dos *solvers* presentes na biblioteca *LEMON*, a mesma utilizada (e desenvolvida) por Kovács (2015) em sua pesquisa. A abordagem foi composta pelo desenvolvimento de um script específico para a manipulação de arquivos *dimacs* e posterior solução da base de dados fazendo uso dos três algoritmos citados previamente e exportação dos resultados organizados em um arquivo .csv. A fim de garantir o registro preciso do tempo de execução de cada algoritmo e evitar a contaminação por possíveis ruídos computacionais, implementamos contadores específicos que registram o tempo de execução somente no intervalo de inicialização e término da execução do solver. Por fim, ocorreu a plotagem de gráficos ilustrando a comparação de desempenho dos algoritmos a partir da planilha criada.

As bases de dados utilizadas no formato *dimacs* são parte das exatas mesmas documentadas no artigo de Kovács (2015). Devido as restrições de hardware da máquina utilizada, a pesquisa foi afunilada a somente 3 bases de dados, Netgen, Gridgen e Goto, todos em sua variante base, desconsiderando as formas _lo e _sr. Cada uma das famílias de bases são compostas por 45 arquivos, sendo divididos em 8 sub-famílias de 5 arquivos cada. Matematicamente, abrangendo o número de nós totais de cada família no intervalo de 2^8 a 2^{16}, resultando em 256 a 65.536 nós em cada sub-família.

A forma escolhida para obtenção do desempenho de cada algoritmo em cada família e subsequente sub-família de bases foi a mesma adotada por Kovács (2015), isto é, soma-se os

resultados de todos os elementos da sub-família e divide-se o resultado pelo número de itens que a compõem, neste caso, 5 itens por sub-família.

Quando nos referimos à estrutura interna de cada família de bases, ambas Netgen e Gridgen possuem dificuldades semelhantes, isto é, relativamente mais fáceis se comparadas com a Goto. A primeira é a forma mais abstrata de rede, por assim dizer, tendo uma média de 8 arcos conectados a cada um dos nós. Ao passo que a rede da família Gridgen, possui uma estrutura interna de grade (tal qual a distribuição de ruas dentro de um bairro específico no qual todas são igualmente espaçadas e divididas em quadrados menores), contando com uma média de 4 arcos conectados a cada nó da rede; nestas duas famílias tanto nós origens quanto destinos aparecem em qualquer ponto da rede. Já na família Goto, a rede gerada é propositalmente complexa e de difícil solução para os algoritmos. Ela possui apenas um nó origem e um nó destino em sua composição e tem os valores de ambos ajustados de acordo com as capacidades dos arcos, o que acaba gerando inevitavelmente gargalos restritos e estressando o algoritmo na busca forçada por vários caminhos alternativos a fim de garantir a viabilidade do fluxo.

RESULTADOS

Em consonância com os resultados obtidos por Kovács (2015), as respostas resultantes da presente pesquisa ficaram dentro dos padrões esperados. No que diz respeito ao tempo total consumido para rodagem das instâncias, a família Netgen levou 71 minutos no total, a Gridgen 9 minutos e a Goto 96 minutos, ressaltando a complexidade expressiva desta última. De acordo com as figuras anexas, o algoritmo que melhor desempenhou em todas as instâncias e bases analisando a partir de um espectro geral foi o *Network Simplex*, sendo ultrapassado apenas nas últimas instâncias da base Goto pelo *Capacity-Scaling Algorithm*. Com base nessas informações, é possível concluir com segurança que para todos os algoritmos analisados, de forma geral, a base mais complexa a ser solucionada foi a Goto.

A explicação para essas métricas variadas reside no funcionamento interno de cada um dos algoritmos analisados em conjunto com a topologia das bases resolvidas.

CONSIDERAÇÕES FINAIS

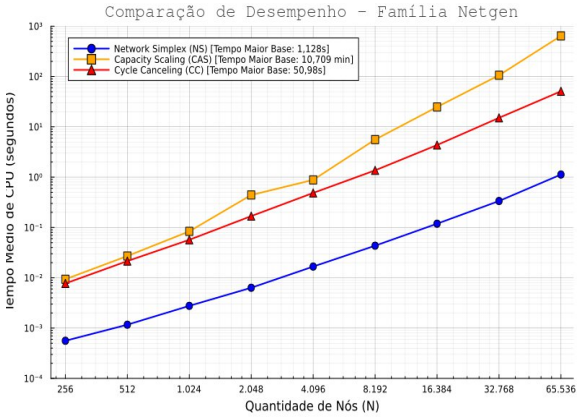
Este estudo comparou de forma prática três abordagens distintas de resolver o problema de fluxo de custo mínimo em redes, reforçando que a velocidade e eficácia de cada método depende diretamente da topologia da rede analisada e da forma particular com a qual o algoritmo soluciona o problema.

Em linhas gerais, o *Network Simplex* provou ser o algoritmo mais rápido em 2/3 das famílias analisadas, sendo elas com conexões aleatórias (Netgen) ou em formato de grade (Gridgen). Contudo, em redes de grande porte com gargalos restritos e caminhos limitados, tal qual a família Goto, conforme a rede vai crescendo em densidade, a melhor opção passa a ser o *Capacity-Scaling Algorithm*. Além disso, a diferença expressiva de tempo de execução registrada no *Simple Cycle Cancelling* entre as redes Gridgen (9 minutos) e Netgen (71 minutos), evidencia que caminhos circulares muito longos atrasam acentuadamente a busca pela otimalidade.

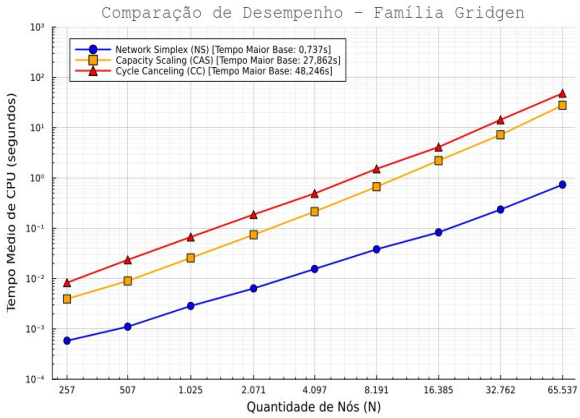
Por fim, os resultados, em convergência com os de Kovács (2015), comprovam que não existe um único algoritmo perfeito para todas as situações. É necessário efetuar uma análise considerando a estrutura interna da rede a ser otimizada para que o melhor algoritmo disponível seja escalado para o trabalho, garantindo o melhor desempenho possível.

Palavras-chave: otimização; pesquisa operacional; benchmark; modelo matemático; fluxo de custo mínimo.

ANEXOS



A) Netgen.



B) Grigen.

Figura 1. *Benchmark Netgen e Gridgen.*

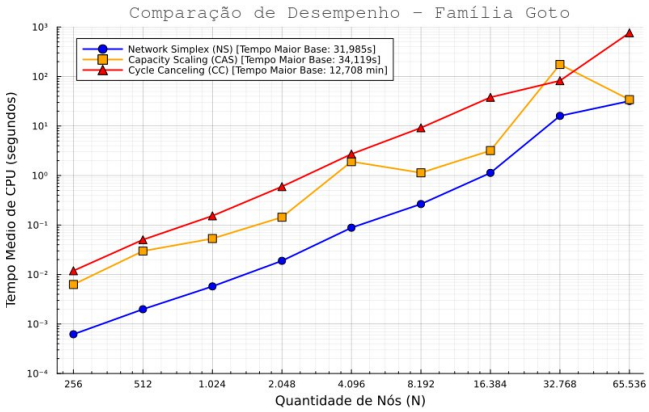


Figura 2. *Benchmark Goto.*

REFERÊNCIAS BIBLIOGRÁFICAS

AHUJA, Ravindra K.; MAGNANTI, Thomas L.; ORLIN, James B. **Network Flows: Theory, Algorithms, and Applications**. Englewood Cliffs, NJ: Prentice Hall, 1993**.**

KOVÁCS, Péter. Minimum-cost flow algorithms: an experimental evaluation. **Optimization Methods & Software**, v. 30, n. 1, p. 94-127, 2015. DOI: 10.1080/10556788.2014.895828.