TRABALHO PRÁTICO:
Single Source Fixed Charge Network Flow Problem

Artur Rodrigues, Luciana Maroun, Thanis Paiva

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

artur@dcc.ufmg.br, lubm@dcc.ufmg.br, thpaiva@dcc.ufmg.br

1. INTRODUÇÃO

O *Single Source Fixed Charge Network Flow Problem* é representado por meio de um grafo contendo apenas um nó de oferta e os demais nós de demanda ou transbordo. Esses nós são conectados através de arcos com capacidades limitadas superiormente e custos fixos não negativos, contabilizados sempre que existir um fluxo em uma aresta, independentemente da intensidade.

Seja um grafo direcionado *G = (N, A)* com vetor de demandas b, de custos c e de limites superiores nos arcos u, todos com valores inteiros, desejamos selecionar um conjunto de arcos $A' \subseteq A$ e atribuir fluxos positivos a esses, de modo que as demandas sejam atendidas, minimizando-se o custo da passagem de fluxo nos arcos e respeitando os limites de capacidade dos mesmos.

Este trabalho prático utiliza diferentes abordagens a fim de resolver o problema descrito. A primeira abordagem consiste na resolução através do solver *GLPSOL*, do pacote *GLPK* (*GNU Linear Programming Kit*), baseado no método *SIMPLEX* para a resolução de problemas de programação linear, e pode ser vista na seção 2.1. A segunda abordagem faz uso de uma heurística que explora as características do problema e foi desenvolvida pela equipe. Detalhes adicionais e a implementação são apresentados na seção 2.2. A partir da geração de instâncias e da efetivação de uma sequência de testes, os métodos são comparados na seção 3.

1.1. Formulação Matemática

Seja o grafo direcionado *G = (N, A)*, mencionado anteriormente, com vetores de demandas b, capacidades u e custos c, desejamos obter um grafo *G' = (N, A')* como definido anteriormente. Porém, antes devemos fazer algumas considerações.

Os nós do grafo *G*, *N*, podem ser divididos entre um nó de oferta, nós de demandas e nós de transbordo. Considerando que $n = |N|$, temos apenas um nó de oferta que será o nó f e os demais $N \setminus \{f\}$ nós serão de demanda ou transbordo.

O nó fonte possui $b_f < 0$, isto é, corresponde ao nó de onde o fluxo sai em direção aos nós de demanda ou transbordo. Assim, $\forall\ i \in N \setminus \{f\}$, $b_i > 0$ (demanda) ou $b_i = 0$ (transbordo). Será assumido que $b_f = -\sum_{i \in N \setminus \{f\}} b_i$, isto é, o somatório da oferta corresponde ao negativo do somatório de todas as demandas do grafo e que $\forall\ (i,j) \in A,\ c_{ij} \geq 0$.

Podemos definir duas variáveis para o problema x_{ij} e y_{ij} tais que:

x_{ij} corresponde ao fluxo no arco $(i,j) \in A$ e

$$y_{ij} = \begin{cases} 1 & \text{se } x_{ij} > 0 \\ 0 & \text{se } x_{ij} = 0 \end{cases}$$

Dessa forma obtemos a seguinte formulação:

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij}$$
$$\sum_{j\in V^-(i)} x_{ij} \; - \; \sum_{j\in V^+(i)} x_{ij} \; = \; b_i, i \; \in \; N$$
$$0 \; \leq \; x_{ij} \; \leq \; u_{ij} y_{ij} \; , (i,j) \; \in \; A$$
$$y \; \in \; \{0,1\}$$

A seguir temos um exemplo de um grafo para o *Single Source Fixed Charge Network Flow Problem*. O grafo contém um nó de oferta, com oferta $b_1 = -20$, nós de demanda com demandas $b_2 = 5$, $b_3 = 7$ e $b_4 = 8$, além de um nó de transbordo com $b_5 = 0$. Os arcos possuem capacidades c_{ij} e limites superiores à passagem de fluxo u_{ij}, representados a seguir como $[c_{ij}, u_{ij}]$.

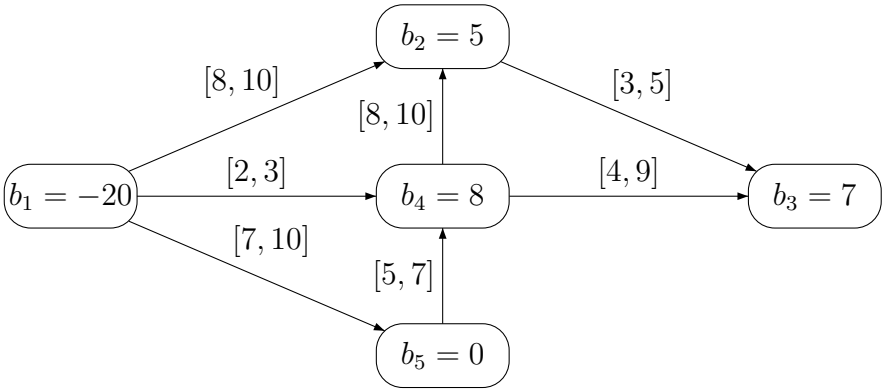

Figura 1. Exemplo de instância para o problema.

2. RESOLUÇÃO DO PROBLEMA

2.1. Solver GLPK

O problema pode ser resolvido pelo GLPK por meio de um modelo descrito pela linguagem GMPL (*GNU Mathematical Programming Language*) e de uma instância que define os parâmetros do problema. O modelo para o problema em questão é apresentado abaixo.

```
/* Conjuntos */
set Nodes;
set Arcs within {Nodes, Nodes};

/* Matrizes e vetores */
param b {Nodes}; /* vetor de demandas */
param c {Arcs}; /* matriz de custos fixos das arestas */
param u {Arcs}; /* matriz de limites superiores das arestas */

/* Variáveis de decisão */
var x {Arcs}, integer; /* fluxo em cada aresta */
var y {Arcs}, binary; /* variável para indicar
                         se fluxo existe em uma aresta */

/* Função objetivo */
minimize objective: sum {(i,j) in Arcs} c[i,j] * y[i,j];
```

```
/* minimizar a soma de custos fixos */

/* Restrições */
s.t. flow_balance {i in Nodes}:
  sum {j in Nodes: (j,i) in Arcs} x[j,i] -
  sum {j in Nodes: (i,j) in Arcs} x[i,j] = b[i];
s.t. capacity {(i,j) in Arcs}:
  x[i,j] <= u[i,j] * y[i,j];
s.t. domain {(i, j) in Arcs}:
  x[i,j] >= 0;

solve; display x;

end;
```

A descrição em GMPL da instância para o exemplo apresentado na figura 1 está representada a seguir,

```
/* Definição do grafo */
set Nodes := 1 2 3 4 5 ;
set Arcs  := (1,2) (1,4) (1,5) (2,3) (4,2) (4,3) (5,4) ;

/* Vetor de demandas dos nós */
param b :=
1     -20
2     5
3     7
4     8
5     0;

/* Vetores de limite superior e de custos,
   respectivamente, das arestas */
param:   u     c :=
1,2      10    1
1,4      3     2
1,5      10    7
2,3      5     3
4,2      8     6
4,3      9     4
5,4      7     5;

end;
```

Para que o GLPK resolva esta instância, basta executar o seguinte comando na linha de comando, em que *modelo.mod* é o arquivo do modelo e *instancia1.dat* é o arquivo da instância:

```
glpsol --model modelo.mod --data instancia1.dat
```

Com isso, o solver reporta sua resolução, informando que a solução para o problema consiste nos seguintes fluxos:

$$x_{12} = 10, x_{14} = 3, x_{15} = 7, x_{23} = 5, x_{42} = 0, x_{43} = 2, x_{54} = 7$$

Conferindo à função objetivo o valor mínimo de 22.

2.2. Heurística do Caminho Mínimo

Para resolver o *Single Source Fixed Charge Network Flow Problem* foi desenvolvida uma heurística que utiliza como base o caminho mínimo entre os nós de demanda e o nó de oferta. Dessa forma, é possível determinar os caminhos de menor custo para suprir todas as demandas, considerando os limites de capacidades dos arcos que compõem tais caminhos.

Partindo do pressuposto de que temos uma instância viável, as demandas possuem um ou mais caminhos – que respeitem os limites da capacidade dos arcos – para o nó de oferta, possibilitando que todas sejam atendidas. Como a resolução do problema visa minimizar o custo total para o atendimento das demandas, isto é, busca minimizar a função objetivo, devemos selecionar o caminho de menor custo entre cada um dos nós de demanda e o nó de oferta. Porém, como os arcos possuem limitações de fluxo, o caminho mínimo encontrado pode não suprir completamente a demanda, de forma que deve-se encontrar outro caminho de custo mínimo para suprí-la.

Considerando que todas as demandas devem ser supridas, a heurística desenvolvida consiste em um método iterativo que a cada iteração seleciona um nó aleatório de demanda com demanda não suprida e calcula o caminho mínimo entre ele e o nó de oferta, utilizando o algoritmo de Dijkstra[Dijkstra 1959], e passa por tal caminho um fluxo capaz de suprir a demanda totalmente ou parcialmente. O atendimento parcial da demanda ocorre quando o caminho mínimo encontrado possui um limite de fluxo inferior ao valor da demanda, fazendo com que o algoritmo tente encontrar outro(s) caminho(s) de custo mínimo para suprí-la totalmente.

Uma vez que um caminho mínimo é obtido, os custos dos arcos que o compõem são contabilizados uma única vez, isto é, caso algum dos arcos faça parte de outros caminhos a passagem de qualquer fluxo adicional não implicará em aumento no custo, já que os custos c_{ij} independem da quantidade de fluxo nos arcos. Dessa forma, paga-se um valor fixo quando um arco é incluído pela primeira vez e, depois de incluído, o fluxo que o percorre não irá provocar alterações no custo já contabilizado anteriormente, possibilitando a relação $c'_{ij} = 0$.

Entretanto, no caso em que em um arco trafega um fluxo máximo, este passa a ter um custo infinito, tornando-o um caminho máximo. Quando, em alguma iteração, ele for obtido como caminho mínimo, corresponde a uma situação de inviabilidade, pois não existem caminhos suficiente para suprir determinado nó de demanda. Quando fluxo imposto não é máximo, isto é, $x_{ij} < u_{ij}$ o limite superior do arco é atualizado e seu novo limite corresponde a $u'_{ij} = u_{ij} - x_{ij}$, possibilitando que o mesmo possa ser utilizado em algum outro caminho.

Em relação aos nós de demanda, caso o primeiro caminho mínimo calculado não seja capaz de atender completamente a demanda, ela deverá ser atualizada, isto é, a nova demanda passa a ser $b'_i = b_i - x_{ij}$ e o nó deverá ter um ou mais novos caminhos mínimos calculados até que a demanda seja completamente suprida, ou seja, $b'_i = 0$.

Em algumas ocasiões raras, durante esse processo iterativo, a heurística pode não achar um caminho por onde passar um fluxo não nulo. Nestes casos, a heurística é reiniciada. A aletoriedade da escolha do nó de demanda faz com que novas soluções sejam exploradas.

A seguir temos o pseudoalgoritmo para a heurística descrita:

1: Heurística do Caminho Mínimo(grafo G)

```
repeat
    Custo_total ← 0;
    foreach arco ∈ A do
        fluxo_arco = 0;
    end
    while existem nós de demanda com a demanda não suprida do
        Seleciona aleatoriamente um nó i com demanda positiva;
        Obtém caminho mínimo do nó i ao nó de oferta;
        limite_superior ← capacidade mínima dentre os arcos do caminho;
        if demanda_i < limite_superior then
            Supre a demanda;
            demanda_i ← 0;
            fluxo ← demanda;
        else
            Demanda parcialmente atentida;
            demanda_i ← demanda_i − limite_superior;
            fluxo ← limite_superior;
        end
        Custo_total ← Custo_total + comprimento do caminho;
        foreach arco do caminho do
            limite_superior_arco ← limite_superior_arco − fluxo;
            if limite_superior_arco = 0 then
                Fluxo máximo no arco, o elimina;
                custo_arco ← infinito;
            else
                Arco pode ser utilizado em outro caminho, custo já pago;
                custo_arco ← 0;
            end
        end
    end
until solução encontrada — tentou mais de 100 vezes ;
```

Considerando demandas, custos e capacidades inteiras, este algoritmo computa o caminho mínimo, no máximo, k vezes para cada nó de demanda, em que k é a demanda máxima de um nó. Isso se deve ao fato de que a demanda decresce de, pelo menos, uma unidade a cada iteração. Considerando a utilização do algoritmo de Dijkstra[Dijkstra 1959] para computar os caminhos mínimos, uma vez que os custos são não negativos, têm-se complexidade assintótica temporal de $k \cdot O(n^2)$. Como o número de nós de demanda está relacionado linearmente com o número total de nós, tem-se complexidade $O(kn^3)$ para a heurística implementada.

O problema apresentado visa à escolha de caminhos de baixo custo e à escolha de poucos caminhos, buscando minimizar o custo que é fixo. Os resultados gerados pela heurística dão preferência ao parâmetro custo dos caminhos em relação ao parâmetro capacidade dos arcos, que potencialmente conferem um menor número de caminhos.

A priorização de caminhos de menor custo não garante uma solução ótima, pois outros caminhos que não sejam mínimos podem comportar um fluxo maior, que no caso da heurística utilizaria mais de um caminho. A escolha do primeiro caminho mínimo faz com que o mesmo seja preferível a todos os outros, pois os custos de suas arestas já foram pagos, mas ele pode não pertencer à solução ótima. Assim, a heurística apresentada segue uma direção de aproximação da solução ótima, via caminhos mínimos, porém sem garantia da otimalidade.

2.3. Implementação da heurística

A heurística foi implementada utilizando-se a linguagem *Python*, com o auxílio da biblioteca *NetworkX*[1] que provê estruturas de dados e funções para o manuseio de Grafos.

Os autores procuraram implementar a heurística de maneira que ela seguisse o mais fielmente o possível o pseudocódigo apresentado nesse documento.

3. EXPERIMENTOS E RESULTADOS

3.1. Instâncias

As instâncias foram obtidas por meio de um gerador que recebe os seguintes parâmetros:

- Número de nós de demanda
- Número de nós de transbordo
- Densidade
- Valor máximo para as demandas
- Valor máximo para as capacidades dos arcos
- Valor máximo para os custos dos arcos

As capacidades, os custos dos arcos e as demandas têm valores máximos definidos como constantes no gerador, e os valores são gerador aleatoriamente respeitando esses limites. O número de nós é dado pela soma dos números de nós de demanda, de transbordo e uma unidade correspondente ao nó único de oferta. Por fim, os arcos são gerados aleatoriamente respeitando-se a densidade inserida.

A viabilidade das instâncias é garantida através do teste de cada uma gerada pelo *solver*. Como ele informa caso a instância seja inviável, aquelas por ele rotuladas dessa maneira são descartadas.

O conjunto de instâncias escolhidas para a análise dos resultados foram divididas em grafos com $10, 20$ e 30 nós. Para cada tamanho do grafo, foi também variada a densidade, com valores 0.25, 0.5, 0.75 e 1. O valor máximo para as demandas nesse conjunto foi de 10, os valor máximo para a capacidade dos arcos foi fixado em 8 e para os custos em 10. Não foi possível utilizar instâncias de tamanho maior, pois o tempo para ser achar a solução ótima através do *GLPSOL* é proibitivo.

Foram também geradas instâncias adicionais com valores máximo para as demandas de 10, 20, 30, 40, 50, 60 e 70. Fixou-se o tamanho das instâncias desse conjunto em 10 nós e a densidade em 0.25.

3.2. Procedimentos

Com o intuito de se obter testes mais consistentes, os experimentos foram executados em ambiente virtualizado, com capacidade de processamento e memória primário reduzidas, 30% da capacidade da máquina hospedeira e 384MiB, respectivamente. O sistema operacional do ambiente virtualizado era Ubuntu 12.04 64 bits e os softwares utilizados foram Python versão 2.7.3, GLPSOL: GLPK LP/MIP Solver versão 4.47 e GCC versão 4.6.3. A máquina hospedeira possuía sistema operacional Mac OS X 10.7.4, processador *quad-core* de 2.3GHz e memória primária com capacidade de 8GiB.

Para cada instância utilizada, foram aferidos o tempo de execução e o valor da função objetivo tanto no *GLPSOL* quanto na implementação da heurística. É importante ressaltar que

[1] http://networkx.lanl.gov/

foram realizadas três execuções e aferidos os valores médios, tanto para o tempo de execução, quanto para o valor da função objetivo (no caso dos procedimentos da implementação da heurística).

3.3. Resultados

A tabela 2 contém os resultados de experimentos realizados através de resoluções de instâncias pelo solver e pela heurística.

		Tempo (s)			Função Objetivo		
Nós	**Dens.**	**GLPK**	**Heurística**	**Aceleração**	**GLPK**	**Heurísitca**	**Distância**
10	0.249	0.1	0.001	172.12	112	128	14.3%
	0.5	0.1	0.001	91.66	259	281	83.7%
	0.749	0.1	0.001	77.34	223	252	56.5%
	1.00	0.1	0.001	73.10	176	174	52.6%
20	0.25	0.2	0.002	92.94	108	146	35.2%
	0.5	8.3	0.006	1,355.10	355	576	62.3%
	0.75	3.3	0.006	524.14	161	301	87.0%
	1.00	5.1	0.010	487.99	94	166	76.6%
30	0.249	10.7	0.008	1,318.06	315	553	75.6%
	0.5	1390.7	0.014	102,611.97	370	578	56.2%
	0.749	453.2	0.017	25,966.88	205	306	49.3%
	1.00	925.7	0.035	26,493,99	212	342	61.3%
Média							47.4%
DPA[2]							20.7%

Tabela 1. Tabela comparativa para diferentes tamanhos e densidades
Valor máximo para as demadas = 10

Nota-se facilmente que a heurística é mais eficiente em termos de tempo de execução do que o *GLPSOL*, obtendo valores para a função objetivo razoáveis, com uma média de 47.4% de distância da solução obtida pela primeira com relação ao ótimo. Além disso, percebe-se que a densidade do grafo está diretamente relacionada ao tempo de execução de ambas as abordagens. Os dados dessa tabela são representados na figura abaixo:

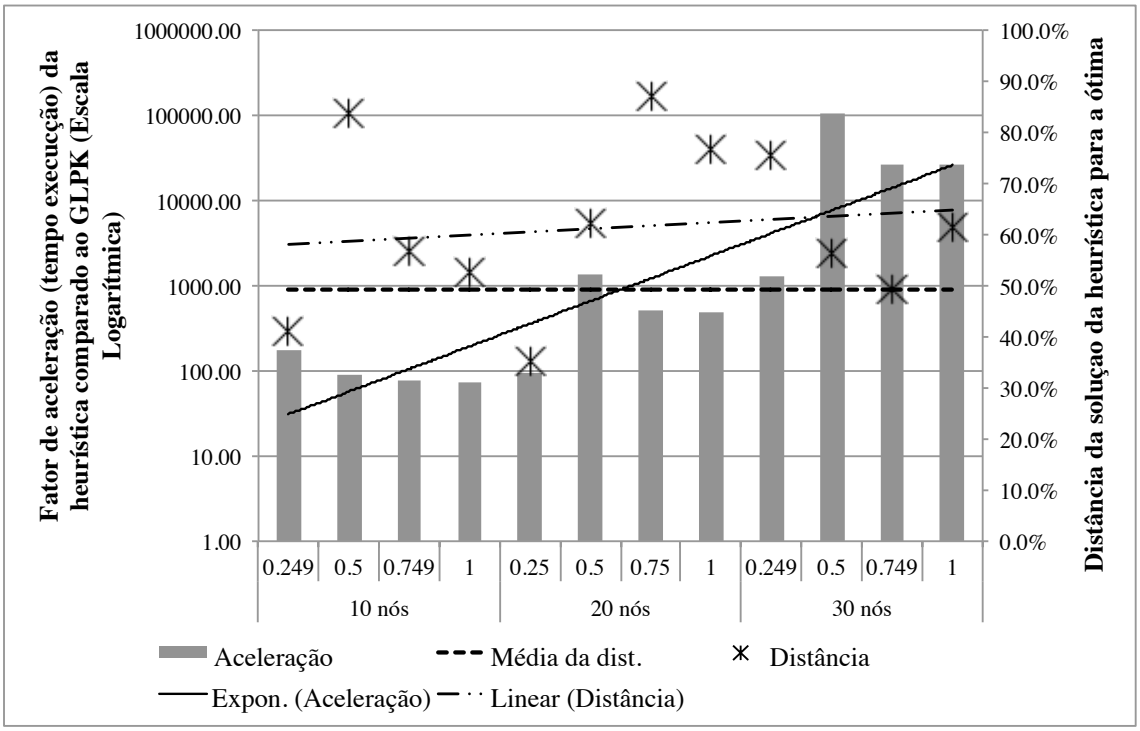

Figura 2. Representação dos dados da tabela 2

É interessante notar que apesar da aceleração, que mede o quão mais rápido a implementação da heurística executou com relação o *GLPSOL*, crescer de maneira exponencial, a distância da solução entre os dois cresce de maneira linear, o que, para alguns cenários de aplicação pode significar um benefício bastante significativo.

Quanto a densidade do grafo, não é possível tirar nenhuma conclusão da influência desse, tanto para a aceleração quanto para a distância.

A tabela abaixo apresenta as mesmas medidas, desta vez, variando-se o valor máximo para as demandas, que é um dos fatores relacionados com a complexidade da heurística proposta. Apesar disso, através das instâncias utilizadas, não foi possível traçar uma relação entre esse valor e a aceleração ou distância. Esse fato pode estar relacionado tanto ao tamanho reduzido dessas instâncias, quanto ao fato dessa valor máximo para as demandas influenciar de maneira decisiva somente no pior caso.

	Tempo (s)			Função Objetivo		
Valor Máx. Dem.	**GLPK**	**Heurística**	**Aceleração**	**GLPK**	**Heurísitca**	**Distância**
10	0.1	0.0007	153.14	51	56	9.8%
20	0.1	0.0005	216.45	40	45	12.5%
30	0.1	0.0008	128.53	39	59	51.3%
40	0.1	0.0006	168.35	77	81	5.2%
50	0.1	0.0005	200.00	45	60	33.3%
60	0.1	0.0006	177.94	46	52	13.0%
70	0.1	0.0005	202.84	49	56	14.3%

Tabela 2. Tabela comparativa para diferentes valores máximos para demanda
Número de nós = 10; Densidade = 0.249

4. CONCLUSÃO

O *Single Source Fixed Charge Network Flow Problem* tem como base um grafo formado por um nó de oferta e nós de demanda ou transbordo conectados através de arcos com limites superiores de capacidade e custos fixos para a passagem de fluxo. Neste trabalho foram utilizadas duas estratégias para resolver o problema descrito acima.

A primeira estratégia corresponde à utilização de um solver, o *GLPSOL* do pacote *GLPK*, que fornece resultados exatos para o problema. A segunda estratégia envolveu a elaboração de um heurística baseada no caminho mínimo entre cada nó demanda e o nó de oferta.

A heurística desenvolvida corresponde a um método iterativo que a cada iteração calcula o caminho mínimo entre o nó de oferta e um nó aleatório de demanda com demanda associada que ainda não tenha sido completamente suprida e passar um fluxo por este caminho. Tal fluxo pode corresponder à capacidade máxima de passagem do caminho ou pode ser inferior a ela. Em ambos os casos a demanda pode ou não ser completamente atendida. Caso a demanda de um dado nó não seja suprida, deve-se calcular outros caminhos mínimos até que ela seja totalmente atendida.

Como a heurística favorece a escolha de caminhos de menor custo, nem sempre seleciona o melhor caminho globalmente, uma vez que um caminho de maior custo poderia permitir uma maior passagem de fluxo e possivelmente diminuir o custo de atendimento da demanda de outros nós. Dessa forma, a heurística não é exata, mas fornece resultados próximos da solução ótima do problema, como comprovado através de experimentos.

Ao final do trabalho foi possível perceber que o *Single Source Fixed Charge Network Flow Problem* apresenta uma formulação simples, mas de difícil solução quando se deseja a solução ótima do problema. Porém, foi possível desenvolver uma heurística relativamente simples que possibilitou calcular soluções razoavelmente próximas da solução ótima.

Referências

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271. 10.1007/BF01386390.

Wolsey, L. A. (1998). *Integer programming*. Wiley-Interscience, New York, NY, USA.

[2]Desvio Padrão Amostral