Um Algoritmo Evolutivo Multiobjetivo em Duas Fases Aplicado ao Problema de Fluxo Multiproduto Binário com Balanceamento da Rede

Fábio P. Mourão, **Sérgio R. de Souza**,
Eduardo G. Carrano,
Programa de Mestrado em Modelagem Matemática e Computacional
Centro Federal de Educação Tecnológica de Minas Gerais
30510-000, Belo Horizonte, MG
E-mail: fabiomourao@terra.com.br, sergio@dppg.cefetmg.br, egcarrano@yahoo.com.br

Marcone J. F. Souza
UFOP - Departamento de Computação
Campus Universitário
35.400-000, Ouro Preto, MG
E-mail: marcone@iceb.ufop.br.

Resumo: *Este trabalho analisa o comportamento de um método evolutivo de duas fases aplicado ao Problema de Fluxo Multiproduto Multiobjetivo. Os objetivos considerados são* (i) *minimizar o custo, característico dos problemas de fluxo multiproduto; e* (ii) *obter o melhor balanceamento possível da rede, procurando homogeinizar o fluxo de produtos pelos arcos. A primeira fase do método possui o intuito de encontrar o número máximo de soluções factíveis para serem usadas na Fase Dois. A Fase Um corresponde ao Algoritmo Genético (AG), utilizando, como função de avaliação, a soma das violações das restrições. A Fase Dois corresponde à aplicação do método evolucionário de Otimização Multiobjetivo NSGA II. Os testes computacionais incidiram sobre instâncias geradas aleatoriamente e mostram a eficiência do método proposto.*

Palavras-chave: *Problemas de Fluxo Multiproduto, Algoritmo Genético, Algoritmos Multiobjetivos*

1 Introdução

Problemas de Fluxo Multiproduto têm sido estudados desde os anos 60, a partir das contribuições iniciais de [4] e [5]. A formulação típica destes problemas consiste em determinar o fluxo de custo mínimo, de forma a atender as restrições de capacidade e de conservação de fluxo da rede, para os vários produtos envolvidos. Portanto, busca-se satisfazer um único objetivo, sem, no entanto, proceder a qualquer análise relativa à estrutura da rede em questão. Há duas estruturas destes problemas. A primeira considera que as variáveis de fluxo são variáveis inteiras. A segunda assume que estas variáveis são binárias e, portanto, o fluxo de produtos é indivisível desde o nó de origem até o nó de demanda. Neste caso, cada produto possui um par de nós origem-destino, correspondentes ao nó de oferta e ao nó de demanda deste produto.

Problemas de fluxo multiproduto são particularmente estudados atualmente no caso de análises de tráfego via internet. Nestas situações, uma consideração de importância é a relativa ao balanceamento de tráfego, com o objetivo de evitar o congestionamento nas redes. Uma aplicação desta classe, voltada, no entanto, apenas para a questão de congestionamento de redes, pode ser encontrada em [6].

Em uma aplicação prática, considerar apenas a função de custo do problema pode levar a possíveis dificuldades na distribuição da rede. Assim, é interessante que se possa apresentar uma formulação que leve em conta tanto a questão de custo mínimo quanto a questão de balanceamento da rede, de modo a evitar a sobrecarga dos arcos da mesma. Um problema desta natureza pode ser posto como um problema de congestionamento mínimo ao custo mínimo, ou seja, trata-se de um problema de balanceamento dos arcos da rede, buscando garantir, ao mesmo tempo, o menor custo para o fluxo pela mesma.

Os objetivos são conflitantes, pois determinar uma solução com menor custo não significa encontrar uma solução com melhor balanceamento. Para entender isso, basta avaliar uma rede tendo um subconjunto de arcos com baixos custos associados, em relação aos demais arcos da rede. Considerando-se apenas a função de custo, sem observar a questão do balanceamento do tráfego, as melhores soluções são, então, aquelas que sobrecarregam os arcos deste subconjunto e subutilizam os arcos de maior custo, levando, portanto, a uma situação de desbalanceamento da rede. Situações como esta justificam, assim, a proposição do estudo do problema de congestionamento mínimo ao custo mínimo.

Este problema, no entanto, apesar de sua clara importância econômica e aplicação efetiva, não foi objeto de consideração na literatura, na forma multiobjetivo aqui proposta, pelo menos de conhecimento dos autores do presente artigo.

2 Formulação do Problema

A formulação matemática do problema multiobjetivo considerado neste trabalho é dada pelas expressões 1 a 6. Será tratado o problema binário, que é um caso particular do problema inteiro. No problema de fluxo multiproduto inteiro, as variáveis precisam ser inteiras, mas o fluxo de cada produto pode ser dividido por diferentes rotas. No problema binário, porém, o fluxo de cada produto deve usar uma única rota.

Seja uma rede $\mathcal{R}$ com n nós, a arcos e p produtos e sejam também $\mathcal{N}$, $\mathcal{A}$ e $\mathcal{P}$, os conjuntos de nós, arcos e produtos, respectivamente. O modelo matemático, então, é dado por:

$$\min \quad F = (f, \Phi) \tag{1a}$$

$$\text{suj. a} \quad \sum_{(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{(j,i)\in\mathcal{A}} x_{ji}^k = b_i^k, \quad \forall\ i \in \mathcal{N}, \quad 1 \leqslant k \leqslant p \tag{1b}$$

$$\sum_{k\in P} x_{ij}^k \leqslant u_{ij}, \quad \forall (i,j)\ \in A \tag{1c}$$

$$x_{ij}^k \in \mathcal{Z}_+ \tag{1d}$$

$$l_{ij} = \sum_{k\in\mathcal{P}} x_{ij}^k, \quad \forall (i,j) \in \mathcal{A} \tag{1e}$$

no qual a expressão (1b) representa as restrições de conservação de fluxo; a expressão (1c) representa as restrições de capacidade; a expressão (1d) representa a restrição de integralidade e de positivadade; x_{ij}^k representa a variável de decisão, que é o fluxo do produto k pelo arco (i,j). Caso este valor seja não-nulo e positivo, significa que o produto faz uso do arco em questão; u_{ij} representa a capacidade do arco (i,j); b_i^k indica se o nó i é oferta/demanda para o produto k. Considerando d^k a oferta/demanda para o produto k, i.e., d^k é a quantidade de produto k que precisa sair de um nó origem e chegar a um nó destino e a cada produto k está associado um par origem-destino, então $b_i^k = d^k$, se o nó i é oferta para o produto k; $b_i^k = -d^k$, se o nó i é demanda para o produto k; e $b_i^k = 0$, se o nó i é um nó de transbordo. l_{ij} representa a carga no arco (i,j), correspondente à soma dos produtos que trafegam no arco; f é a função de custo do problema de fluxo multiproduto, dada por:

$$f = \sum_{k=1}^{p} \sum_{(i,j)\in A} c_{ij}^k x_{ij}^k \tag{2}$$

para c_{ij} correspondente ao custo para o produto k trafegar pelo arco (i, j).

A taxa de utilização t_{ij} de um arco (i, j) é definida como:

$$t_{ij} = \frac{l_{ij}}{u_{ij}} \tag{3}$$

Para medir o congestionamento total da rede, é preciso, primeiro, encontrar o custo do congestionamento em cada arco, representado pela função $\Phi_{ij}(l_{ij})$, de forma que um arco com maior taxa de utilização é mais penalizado que outros com menores taxas. Assim,

$$\Phi = \sum_{(i,j)\in\mathcal{A}} \Phi_{ij}(l_{ij}) \tag{4}$$

A diferencial da função Φ_{ij} em relação a l_{ij} é dada, então, por [7]:

$$\frac{\partial\Phi_{ij}(l_{ij})}{\partial l_{ij}} = \begin{cases} 1, \text{ se } 0 \leq l_{ij} < (1/3)u_{ij}; \\ 3, \text{ se } (1/3)u_{ij} \leq l_{ij} < (2/3)u_{ij}; \\ 10, \text{ se } (2/3)u_{ij} \leq l_{ij} < (9/10)u_{ij}; \\ 70, \text{ se } (9/10)u_{ij} \leq l_{ij} < 1.u_{ij}; \\ 500, \text{ se } 1.u_{ij} \leq l_{ij} < (11/10)u_{ij}; \\ 5000, \text{ se } l_{ij} \geq (11/10)u_{ij}. \end{cases} \tag{5}$$

Estes valores foram obtidos empiricamente e são também utilizados no presente trabalho. A função $\Phi_{ij}(l_{ij})$ é dada, então, segundo [7], por:

$$\Phi_{ij}(l_{ij}) = \begin{cases} l_{ij}, \text{ se } 0 \leq l_{ij} < (1/3)u_{ij}; \\ 3l_{ij} - (2/3)u_{ij}, \text{ se } (1/3)u_{ij} \leq l_{ij} < (2/3)u_{ij}; \\ 10l_{ij} - (16/3)u_{ij}, \text{se } (2/3)u_{ij} \leq l_{ij} < (9/10)u_{ij}; \\ 70l_{ij} - (178/3)u_{ij}, \text{ se } (9/10)u_{ij} \leq l_{ij} < u_{ij}; \\ 500l_{ij} - (1468/3)u_{ij}, \text{ se } u_{ij} \leq l_{ij} < (11/10)u_{ij}; \\ 5000l_{ij} - (16381/3)u_{ij}, \text{ se } l_{ij} \geq (11/10)u_{ij}. \end{cases} \tag{6}$$

Esta função, então, penaliza cada arco de acordo com sua taxa de utilização, de modo que arcos próximos de serem violados, ou violados, apresentam um maior valor no coeficiente angular na função supracitada, que é convexa e linear por partes. Foram utilizados os mesmos intervalos e valores de derivadas utilizados por [7]. Para cada intervalo, é atribuído uma valor diferencial diferente, a fim de penalizá-lo.

3 Algoritmo NSGA II

O método NSGA (*Nondominated Sorting Genetic Algorithm*) foi proposto em [2], sendo um dos primeiros algoritmos evolucionários propostos para a solução de problemas de otimização multiobjetivo. Contudo, de acordo com [1], este método é caro computacionalmente para grandes tamanhos de populações, pois possui complexidade computacional de ordenamento por não-dominância igual a $\mathcal{O}\left(MN^3\right)$, sendo M o número de objetivos e N o tamanho da população. Além disso, utiliza uma abordagem não-elitista, o que pode levar à perda de boas soluções. Outra dificuldade diz respeito há necessidade de especificar um parâmetro de compartilhamento, utilizado para manter a diversidade na população, sendo este um parâmetro de difícil determinação. O método NSGA II, proposto em [1], busca reduzir o tempo computacional; tornar o método elitista; e, também, melhorar o método evolutivo quanto a diversidade, ultrapassando, assim, as dificuldades existentes na versão anterior.

Duas questões são fundamentais neste método: ordenação de soluções não-dominadas e diversidade da população.

Ordenamento de soluções não-dominadas O algoritmo de ordenação de soluções não-dominadas e que cria as fronteiras de soluções no algoritmo é descrito sucintamente a seguir. O método NSGA II encontra, primeiro, para cada solução s, o número de soluções que dominam essa solução, denotado por n_s (contador de dominância) e o conjunto S_s, formado pelo número de soluções dominadas por s. De acordo com [1], isso requer $O(MN^2)$ comparações. Assim, todas as

```
fast-non-dominated-sort(P)
para cada p ∈ P
    S_p = ∅
    n_p = 0
    para cada q ∈ P
        se (p ≺ q) //se p domina q
            S_p = S_p ∪ {q} //adiciona q ao conjunto de soluções dominadas
por p
        senão se (q ≺ p)
            n_p = n_p + 1 //incrementa o contador de dominância de p
    se (n_p = 0)
        p_rank = 1 //insere p no front 1
        F_1 = F_1 ∪ {p}
i = 1 //inicializa o contador de front
Enquanto F_i ≠ ∅
    Q = ∅
    para cada p ∈ F_i
        para cada q ∈ S_p
            n_q = n_q − 1
            se (n_q = 0)
                q_rank = i + 1 //neste caso q pertence ao próximo front
                Q = Q ∪ {q} //insere q no outro nível de dominância
    i = i + 1
    F_i = Q
fim fast-non-dominated-sort(P)
```

Figura 1: Pseudo-código *Fast non Dominated Sort.*

soluções não-dominadas possuem $n_s = 0$ e formarão o primeiro *front*. Agora, para cada solução do *front* 1, cada solução q pertencente ao conjunto S_s é visitada e o contador de dominância n_q é decrementado em uma unidade. Os elementos q com contador de dominância iguais a zero são, então, separados para uma lista Q e estes elementos pertencentes à lista Q pertencem, também, ao segundo nível de não-dominância, ou seja, pertencem ao *front* 2. O processo é repetido para que se determine os demais níveis de não-dominância. Em [1], é demonstrado que a complexidade computacional deste método é dada por $O(MN^2)$. Este método de ordenação por não-dominância é conhecido como *Fast non Dominated Sort* e seu pseudo-código, de acordo com [1], é apresentado na Figura 1, .

Diversidade da população de soluções No algoritmo NSGA II, o método de compartilhamento utilizado para manter a diversidade da população de soluções é através de uma *crowded-comparison*, que não possui necessidade de especificar nenhum parâmetro de compartilhamento e possui uma melhor complexidade computacional. A função *Crowding distance* (distância de agrupamento) é definida como sendo o perímetro do cubóide criado a partir de um ponto, tendo como vértices os vizinhos deste ponto. Só há sentido em calcular seu valor entre as soluções de um mesmo *front*. As soluções extremas dos *fronts*, que possuem apenas um vizinho, recebem um valor infinito e sempre terão preferência no método elitista. Na Figura 2, apresentada inicialmente em [1], é exibida uma representação gráfica do que vem a ser a *crowding distance* de uma solução i em um *front*. O cálculo da função *crowding distance* necessita do ordenamento em ordem crescente dos objetivos. Em seguida, cada solução extrema do *front* recebe um valor arbitrariamente grande. Todas as soluções intermediárias receberão um valor igual à diferença absoluta normalizada em função dos valores das duas soluções adjacentes pertencentes ao mesmo *front*. O cálculo é repetido para cada objetivo. O valor global de cada solução corresponde à soma dos valores em relação a cada objetivo, calculado da forma supracitada. Antes de calcular a função *crowding distance*, cada objetivo é normalizado. O algoritmo para calculo da *crowding distance* é apresentado pela Figura 3, sendo $\mathcal{I}$ um conjunto de soluções de um mesmo nível de não-dominância.

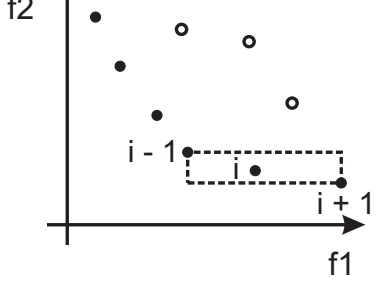

Figura 2: *Crowding distance.*

crowding-distance($\mathcal{I}$)
$l = |\mathcal{I}|$ //número de soluções em $\mathcal{I}$
para cada $i \in \mathcal{I}$, $\mathcal{I}[i]_{\text{distância}} = 0$ //incializa valor da crowding distance para cada solução i
para cada objetivo m
$\mathcal{I}$ = ordena($\mathcal{I}$,m) //ordena o front em relação a cada função objetivo
$\mathcal{I}[1]_{\text{distância}} = \mathcal{I}[\ l\ ]_{\text{distância}} = \infty$ //atribui um valor arbitrariamente grande às soluções extremas do front
para cada i = 2 até l - 1)
$\mathcal{I}[i]_{\text{distância}} = \mathcal{I}[i]_{\text{distância}} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)/(f_m^{\max} - f_m^{\min})$
fim

Figura 3: Pseudo-código - *Crowding Distance*

4 Metodologia

Nesta seção serão discutidas algumas características da metodologia computacional adotada em ambas as fases do método proposto. Como já mencionado anteriormente, o método constituiu-se de dois algoritmos evolutivos, sendo o da Fase Um um Algoritmo Genético (AG) mono-objetivo, com o intuito de minimizar o somatório das violações das restrições de capacidade. Portanto, a função de aptidão do AG na primeira fase é dada por:

$$g = \sum_{(i,j)\in A} \nu_{i,j} \tag{7}$$

sendo $\nu_{i,j} = 0$, se o arco (i, j) não for violado; e $\nu_{i,j} = (\sum_{k=1}^{p} x_{i,j}^k) - u_{i,j}$, se o arco (i, j) for violado. Já na Fase Dois são considerados os dois objetivos descritos anteriormente.

A representação de uma solução é feita por meio de uma matriz X de dimensão $a \times p$. Em vez de serem atribuído valores 0 ou 1 a cada posição dessa matriz, é atribuído o valor que representa a quantidade do produto que passa em cada posição. Já a população inicial na Fase Um é gerada por meio de uma heurística aleatória desenvolvida especificamente para o problema. Na Fase Dois (NSGA II), a população é gerada por meio da execução da Fase Um. A seleção de indivíduos para a recombinação é feita de forma aleatória entre todos os indivíduos da população, com probabilidade uniforme, em ambas as fases. Foi utilizado o operador *crossover* uniforme nas duas fases. Uma descrição detalhada deste tipo de operador pode ser encontrada em [8]. O operador de Mutação na Fase Um pode ser determinístico ou aleatório, ou seja, com dada probabilidade, pode ser executado um algoritmo de busca local na nova solução gerada ou então podem ser trocadas algumas colunas na matriz solução. Foge do escopo deste trabalho descrever tal método de busca. Na Fase Dois, a mutação é aleatória, dada certa probabilidade. Por fim, para a seleção de indivíduos na Fase Um, é usado um simples método elitista, mantendo os melhores indivíduos, enquanto na Fase Dois utiliza-se a natureza elitista do NSGA II.

3 Resultados

Nesta seção são apresentados os gráficos das soluções candidatas a Pareto-ótimas para cada instância testada. Os testes computacionais incidem sobre instâncias geradas aleatoriamente pelo GenMCF, desenvolvido por [3] e realizados em um computador Intel Celeron M 1.87 GHz, com 1 GB de RAM DDR 2 sob o sistema Operacional Windows XP SP 2. Os algoritmos foram implementados utilizando a linguagem de programação C, sendo utilizado o compilador Borland C++ Builder 5.0. Os gráficos foram gerados por meio do MatLab 6.5.

As instâncias testadas e os tempos computacionais, em segundos, são apresentados pela Tabela 1. Na Tabela 1, $\#N$, $\#A$ e $\#P$, correspondem ao número de nós, arcos e produtos, respectivamente. A coluna $\#P/\#A$ indica a densidade da rede, em termos da relação entre o número de produtos e o número de arcos.

Tabela 1: Instâncias Testadas.

Instância	#N	#A	#P	#P/#A	Tempo (s)
bl01	32	96	48	0,5	1493,265
bl02	32	96	48	0,5	2068,828
bl03	32	96	48	0,5	1639,125
bl05	32	320	48	0,15	1305,985
bl07	32	320	48	0,15	1114,265
bl11	32	96	192	2	7595,094
bl12	32	96	192	2	9272,703

Os valores obtidos foram normatizados entre 0 e 1 e representados pelos gráficos apresentados nas Figuras 4-10, nas quais o eixo das abscissas representa os valores das funções de custo e o eixo das ordenadas representa os valores da funções de congestionamento.

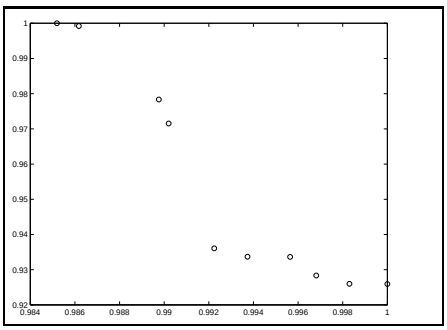

Figura 4: Instância bl01.

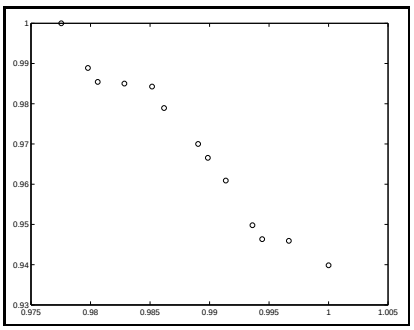

Figura 5: Instância bl02.

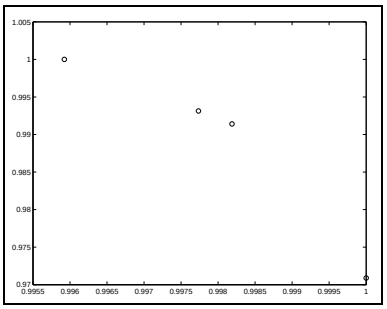

Figura 6: Instância bl03.

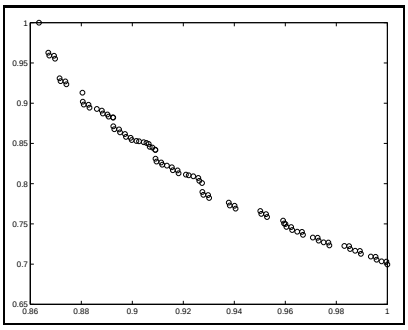

Figura 7: Instância bl05.

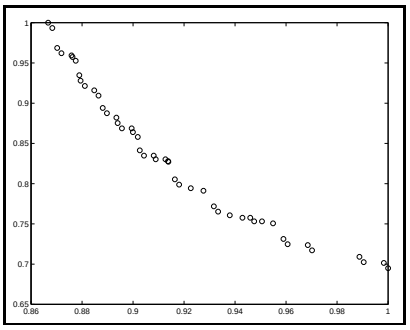

Figura 8: Instância bl07.

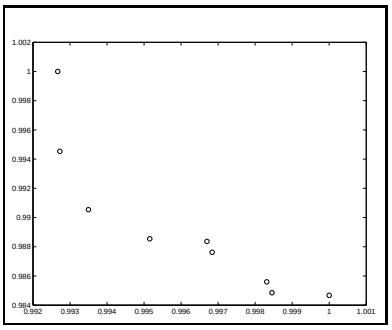

Figura 9: Instância bl11.

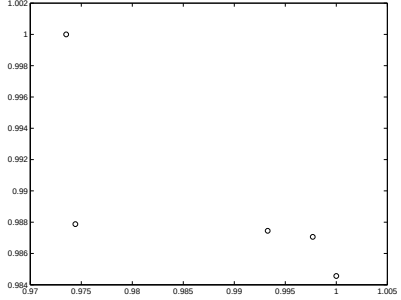

Figura 10: Instância bl12.

6 Conclusões

O método proposto se mostrou eficiente, encontrando soluções candidatas a Pareto-ótimas para as instâncias testadas. Em todos os casos, um melhor balanceamento na rede implica em um pior valor na função de custo, fato que mostra que os objetivos tratados são conflitantes. Os resultados foram satisfatórios e, como já foi dito, infelizmente não foram feitas comparações com outros resultados da literatura, pois não foram encontrados, na pesquisa bibliográfica realizada, trabalhos com a mesma abordagem. As instâncias cuja relação entre a quantidade de produtos pela quantidade de arcos foram menores obtiveram um maior número de soluções candidatas.

Referências

[1] K. Deb, S. Agrawal, A. Pratab, T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6:2 (2002) 182 - 197.

[2] N. Srinivas, K. Deb, Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, *Evolutionary Computation*, 2:3 (1994) 221-248.

[3] F. P. Alvelos, "Branch-And-Price and Multicommodity Flows", Tese de Doutorado, Universidade do Minho, Portugal, 2005.

[4] L.R. Fulkerson, D.R. Ford, "Flows in Networks", Princeton University, USA, 1962.

[5] T.C. Hu, Multicommodity Network Flows, *Operations Research.*, 11 (1963) 344-360.

[6] L. S. Buriol, "Roteamento do Tráfego na Internet: algoritmos para projeto e operação de redes com protocolo OSPF", Tese de Doutorado, UNICAMP, 2003.

[7] B. Fortz and M. Thorup, Increasing internet capacity using local search, em "IEEE Conf. on Computer Communications (INFOCOM)", 2000.

[8] G. Syswerda, Uniform Crossover in Genetic Algorithms, em "Proceedings of the Third International Conference on Genetic Algorithms" pp. 2-9, J.D., Morgan Kaufmann Publishers, 1989.