UM ALGORITMO EXATO PARA UM PROBLEMA DE PROGRAMAÇÃO INTEIRA BIOBJETIVO

Leizer de Lima Pinto
Instituto de Informática - UFG
Rua Alameda Palmeiras, Quadra D, Câmpus Samambaia - CEP 74690-900 - Goiânia - GO
leizer@inf.ufg.br

Kátia Cilene Costa Fernandes
Instituto de Informática - UFG
Rua Alameda Palmeiras, Quadra D, Câmpus Samambaia - CEP 74690-900 - Goiânia - GO
katiacilene@inf.ufg.br

RESUMO

Este artigo apresenta um problema de programação inteira biobjetivo com aplicações em redes e transportes. Uma função objetivo é do tipo totalizadora (min-soma) e a outra é do tipo gargalo (min-max). Um método exato baseado na técnica ε-restrito é apresentado para gerar um conjunto mínimo completo de soluções Pareto-ótimas, o qual foi implementado em linguagem C++. Prova de otimalidade, uma aplicação envolvendo um problema de transporte e resultados computacionais avaliando o desempenho do método através do solver CPLEX também são apresentados.

PALAVRAS CHAVE. Programação Inteira Biobjetivo, Função Gargalo, Solução Pareto-ótima.

TÓPICOS. ADM - Apoio à Decisão Multicritério, PM - Programação Matemática, OC - Otimização Combinatória.

ABSTRACT

This paper presents a biobjective programming problem with applications in networks and transportations. One objective function is of cost type (min-sum) and the other is of bottleneck type (min-max). An exact algorithm is developed to generate a minimal complete set of Pareto-optimal solutions, which it was implemented in C++. Optimality proof of the algorithm, an application involving a transportation problem and computational experiments evaluating the performance of the algorithm by CPLEX solver are also presented.

KEYWORDS. Biobjective Integer Programming, Bottleneck Function, Pareto-optimal Solution.

PAPER TOPICS. ADM - Multicriteria Decision Support, PM - Mathematical Programming, OC - Combinatorial Optimization.

1 Introdução

O século 21 vive um cenário de intensos gargalos tanto na rede de transporte, pelo crescimento infrene da frota automobilística e o não desenvolvimento das vias rodoviárias, como na rede de comunicação, pelo desenvolvimento significativo dos equipamentos e das ferramentas de tecnologias eletrônicas. Neste sentido, o estudo e o desenvolvimento de técnicas de otimização, tratando esses objetivos, se faz necessário.

Os problemas de fluxos em redes com múltiplos objetivos representam uma classe de problemas da otimização combinatória multiobjetivo e possuem uma extensa variedade de problemas de interesses práticos, por exemplo, em redes de comunicação e sistemas de transporte. Para um estudo aprofundado sobre esses problemas de fluxos em redes e problemas de otimização combinatória multiobjetivo veja [Ahuja et al., 1993; Ehrgott e Gandibleux, 2002; Ulungu e Teghem, 1994]. Muitos desses problemas de fluxos são tratados com programação inteira multiobjetivo (PIMO) [Klein e Hannan, 1982; Clímaco et al., 1997; Alves e Costa, 2012].

A literatura traz vários métodos para determinar todas as soluções eficientes, ou um subconjunto pré-definido delas, para os problemas de PIMO, denominados de métodos geradores. A técnica mais utilizada para achar soluções eficientes é a escalarização, empregada em quase todos os métodos exatos e em muitas técnicas heurísticas, que transforma o problema de PIMO em problemas com um único objetivo, os quais são resolvidos repetidamente. Dentre esses métodos destacam-se o método das restrições (ε-restrito). Ele cria um problema mono-objetivo, através do problema original, mantendo apenas uma das funções objetivo e transformando as demais em restrições. Veja [Ehrgott, 2006].

Em [Hansen, 1980] são apresentados algoritmos de rotulagem, que trabalham diretamente sobre o grafo, para diversos problemas de caminho bicritérios, dentre eles o problema com uma função objetivo totalizadora e outra gargalo. Posteriormente, [Clímaco e Martins, 1982] apresentaram um problema genérico com duas funções objetivo totalizadora e um método para o problema, que classifica os caminhos mínimos, visando determinar os caminhos não-dominados. Dois anos depois, [Martins, 1984] manifestou o interesse em problemas de caminho bicritérios em que uma das funções é do tipo gargalo. Um algoritmo para determinar um conjunto mínimo completo de caminhos não-dominados é apresentado.

Problemas bicritérios em grafos generalizados, que são modelados com duas medidas de desempenhos, sendo uma função totalizadora e a outra uma função gargalo, foi apresentado por [Berman et al., 1990]. O artigo traz três algoritmos com o intuito de achar todas as soluções Pareto-ótimas, com base no método da restrição.

Para o problema de caminho tricritério, com uma função custo e duas funções gargalo, [Pinto et al., 2009] apresentaram o primeiro algoritmo exato e polinomial. O método consiste em determinar caminhos mínimos em subredes, obtidas por um conjunto restrito de arestas de acordo com limites para as funções gargalo. No ano seguinte, [Pinto e Pascoal, 2010] desenvolveram um algoritmo com melhor desempenho computacional, para o mesmo problema. Em 2012, [Bornstein et al., 2012] generalizaram e estenderam o algoritmo anterior para problemas de otimização combinatória multiobjetivo, com uma função totalizadora e n funções gargalo.

Os algoritmos multiobjetivos, acima mencionados, são limitados a resolver problemas em que os pesos, nos elementos que compõem as soluções (por exemplo as arestas do grafo nos problemas de caminho), são fixos. Isto deve ao fato de que eles trabalham com uma ordenação nos pesos associados às funções objetivo de gargalo e, em cada iteração, definem e resolvem um problema mono-objetivo incluindo ou excluindo elementos, com base em tal ordenação.

Em diversos problemas em redes é comum ter como meta o roteamento de fluxos onde, para cada fluxo, é destinado um caminho. Para estes casos, um objetivo clássico é o balanceamento de carga, que consiste em rotear os fluxos, minimizando o gargalo da rede. Outro objetivo comum, em tais problemas, consiste em minimizar o número de saltos para o roteamento dos fluxos. Nestes

problemas, os pesos associados aos elementos (links) passam a ser dinâmicos, pois consistem dos fluxos que serão atribuídos aos links. Uma aplicação para este problema podem ser encontrado em [Gálvez e Ruiz, 2013] ou [De Melo et al., 2014]. Esses trabalhos apresentam algoritmos heurísticos para variantes desse problema.

Neste trabalho, um modelo de programação inteira biobjetivo é apresentado e um método de solução é proposto para resolver esse modelo de forma exata. O método proposto se baseia na técnica ε-restrito e trabalha de forma interativa, resolvendo, em cada iteração, um subproblema mono-objetivo de programação inteira. A função objetivo destes problemas mono-objetivo consiste em minimizar o número total de saltos para o roteamento dos fluxos e, por restrições, lidamos com o gargalo da rede, que é dado pelo valor da carga da(s) aresta(s) mais sobrecarregada(s). Esse método obtém um conjunto mínimo completo de soluções Pareto-ótimas.

Na seção seguinte é apresentado o modelo para o problema, acompanhado de definições e notações utilizadas nesse trabalho. Uma aplicação é exibida nessa mesma seção. Na terceira seção é apresentado o método, juntamente com um exemplo que ilustra o seu funcionamento e provas de otimalidade. A seção 4 traz e discute os resultados computacionais. As considerações finais são exibidas na última seção.

2 O Problema

Considere um conjunto de fluxos F, com $|F| = r$, a ser roteado em um grafo orientado $G = (V,E)$, onde V representa o conjunto de nós e E o conjunto de arestas, com $|V| = n$ e $|E| = m$. Cada fluxo $f \in F$ deve ser roteado por um único caminho em G, indo da sua origem $s_f \in V$ ao seu destino $d_f \in V$. Para cada aresta $e_{sd} \in E$, indo de $s \in V$ para $d \in V$, e fluxo $f \in F$, definimos uma variável binária $x_{f(sd)}$, que irá indicar se o fluxo f vai passar (ou não) pela aresta e_{sd}.

Assim, o modelo de programação inteira biobjetivo, que formulamos para o problema, de rotear os r fluxos, minimizando o gargalo da rede e o número total de saltos, segue-se abaixo.

$$(P) \quad \textit{minimizar} \left\{ \max_{e_{sd} \in E} \left\{ \sum_{f \in F} x_{f(sd)} \right\} \right\} \tag{1}$$

$$\textit{minimizar} \left\{ \sum_{f \in F} \sum_{e_{sd} \in E} x_{f(sd)} \right\} \tag{2}$$

sujeito a:

$$\sum_{e_{s_f d} \in E} x_{f(s_f d)} = \sum_{e_{sd_f} \in E} x_{f(sd_f)} = 1, \; \forall f \in F \tag{3}$$

$$\sum_{e_{sd} \in E} x_{f(sd)} - \sum_{e_{ds} \in E} x_{f(ds)} = 0, \; \forall s \in V - \{d_f, s_f\}, \forall f \in F \tag{4}$$

$$x_{f(sd)} \in \{0,1\}, \; \forall e_{sd} \in E \text{ e } \forall f \in F \tag{5}$$

Enquanto (1) busca minimizar a carga da(s) aresta(s) mais congestionada(s), (2) busca minimizar o total de saltos para o roteamento dos fluxos. As duas restrições em (3) garantem que cada fluxo $f \in F$ saia da sua origem, s_f, e alcança o seu destino, d_f. Além disso, as restrições (3), (4) e (5) garantem, juntas, que cada fluxo segue apenas por um único caminho entre sua origem e seu destino.

Temos que (1) é dita uma função gargalo e (2) uma função totalizadora. Podemos observar que estas duas funções objetivo são conflitantes, pois minimizar a carga da(s) aresta(s) mais congestionada(s) deve implicar em caminhos mais longos para os fluxos. Da mesma forma, minimizar o número total de saltos deve levar a um maior valor para a função gargalo. Claro, dependendo da instância, podemos ter caminhos mais curtos e disjuntos para os fluxos. Isto é, para estes casos existe uma solução com no máximo um fluxo em cada aresta, e o número total de saltos é mínimo, ou seja, ela minimiza as duas funções ao mesmo tempo. Porém, estes casos são raros.

Sejam X, o *conjunto viável* de (P), e $z(x) = [z_1(x), z_2(x)]$, o vetor objetivo associado a $x \in X$, onde:

$$z_1(x) = \max_{e_{sd} \in E} \sum_{f \in F} x_{f(sd)} \quad \text{e} \quad z_2(x) = \sum_{f \in F} \sum_{e_{sd} \in E} x_{f(sd)}.$$

Para $x, x' \in X$, dizemos que x *domina* x' quando $z(x) \leq z(x')$ e $z(x) \neq z(x')$. Assim, uma solução viável $x \in X$ é uma solução *Pareto-ótima* se não existe nenhuma outra solução em X que domine x.

O conjunto $X^* \subseteq X$, formado por soluções Pareto-ótimas, é um *conjunto mínimo completo* quando: 1) para cada par de soluções $x^*, \bar{x} \in X^*$ temos $z(x^*) \neq z(\bar{x})$; e, 2) para qualquer solução Pareto-ótima $x \in X$ existe $x^* \in X^*$ tal que $z(x) = z(x^*)$.

Uma aplicação. Suponha uma via rodoviária representada por um grafo onde, cada trecho (aresta) ligando duas cidades (nós), exista um ponto de parada (fiscalização obrigatória ou pedágio). A função z_2 pode representar o total de paradas dos caminhões, de uma determinada companhia de transporte, que trafegam por esta via. A função z_1 visa medir a densidade do tráfico no trecho mais congestionado. A companhia pode estar interessada em, simultaneamente, minimizar a quantidade total de paradas dos seus caminhões e balancear a carga na via, minimizando o gargalo.

3 O Algoritmo

Nesta seção é apresentado um método biobjetivo para a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas para o problema (P). Em cada iteração um subproblema mono-objetivo é definido e resolvido, o qual é um problema de programação inteira 0-1, denominado por (P_α).

A função objetivo de (P_α) é dada por z_2, isto é, a função totalizadora (2) do problema biobjetivo (P). As restrições de (P_α) são (3), (4) e (5) de (P), juntamente com uma restrição sobre o gargalo, a qual é definida através do parâmetro α.

$$(P_\alpha) \quad \textit{minimizar} \left\{ \sum_{f \in F} \sum_{e_{sd} \in E} x_{f(sd)} \right\}$$

sujeito a:

$$\sum_{e_{s_f d} \in E} x_{f(s_f d)} = \sum_{e_{s d_f} \in E} x_{f(s d_f)} = 1, \ \forall f \in F$$

$$\sum_{e_{sd} \in E} x_{f(sd)} - \sum_{e_{ds} \in E} x_{f(ds)} = 0, \ \forall s \in V - \{d_f, s_f\}, \forall f \in F$$

$$\sum_{f \in F} x_{f(sd)} \leq \alpha, \ \forall e_{sd} \in E$$

$$x_{f(sd)} \in \{0,1\}, \ \forall e_{sd} \in E \text{ e } \forall f \in F$$

A escolha de manter a função totalizadora (2) como função objetivo, e tratar a função gargalo (1) como restrição, se deve ao fato de (2) ser uma função linear e (1) não ser. Porém, as restrições associadas ao parâmetro α, que limitam os valores de (1), são lineares. Logo, esta estratégia nos permite resolver o problema (P) por programação linear.

Inicialmente fazemos $\alpha = \infty$, e após a obtenção da solução ótima $\bar{x}$ para (P_α), fazemos $\alpha = z_1(\bar{x}) - 1$. Desta forma, o gargalo da nova solução, $\bar{x}$, obtida na iteração seguinte, será menor do que o da anterior, renomeada para $\hat{x}$. Então, caso o valor de z_2 seja o mesmo para estas duas soluções, temos que a solução anterior é dominada pela nova solução, daí a solução anterior é deletada. Assim, segue o algoritmo até uma iteração em que (P_α) seja inviável. O algoritmo completo é apresentado a seguir.

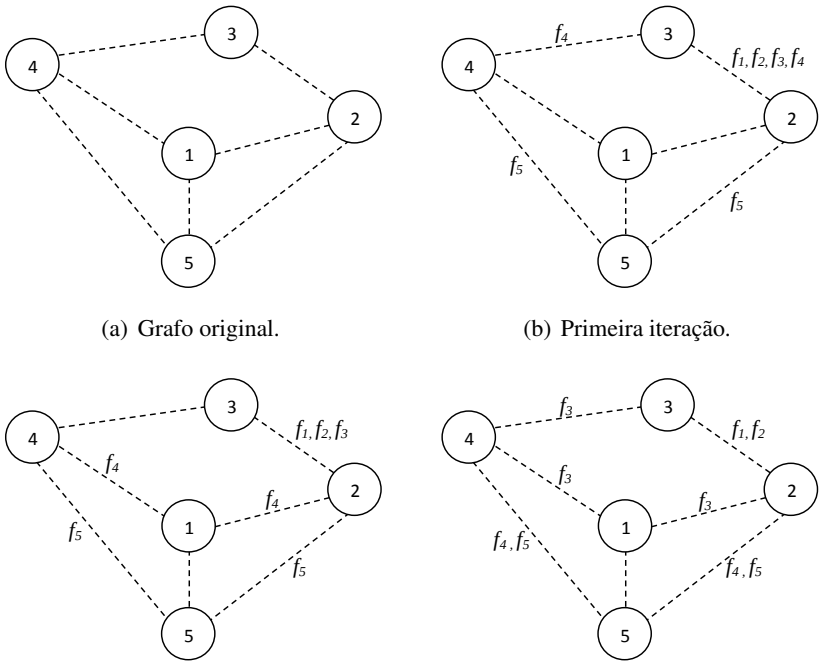
Algoritmo 1.

1. $X^* \leftarrow \emptyset$; $\alpha \leftarrow \infty$.
2. **Enquanto** (P_α) *não for inviável* **faça**
3. $\quad \bar{x} \leftarrow$ *solução ótima de* (P_α)
4. $\quad X^* \leftarrow X^* \cup \{\bar{x}\}$
5. $\quad \alpha \leftarrow z_1(\bar{x}) - 1$
6. $\quad$ **Se** $|X^*| > 1$ **e** $z_2(\bar{x}) = z_2(\hat{x})$ **então**
7. $\quad\quad X^* \leftarrow X^* - \{\hat{x}\}$
8. $\quad \hat{x} \leftarrow \bar{x}$

O decremento de exatamente uma unidade no valor de α, de uma iteração para a próxima, é necessário para garantir a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas. Como os valores de z_1 são inteiros, a atualização de α garante que qualquer solução, com z_1 menor do que o da última solução obtida, será viável para o subproblema atual. Porém, embora α sofra esse pequeno decréscimo, a solução obtida pode ter um valor para z_1 menor do que este limite. Por exemplo, se em uma dada iteração temos $\alpha = 15$ e obtemos uma solução com $z_1 = 10$, na iteração seguinte o limite passa a ser $\alpha = 9$.

Temos que a matriz tecnológica dos problemas (P_α) é totalmente unimodular. Desta forma, a resolução da relaxação linear destes problemas pelo algoritmo Simplex, quando estes admitem solução ótima, resultará em uma solução ótima inteira. Isto acelera a resolução de (P) pelo Algoritmo 1. Além disso, podemos observar que de um (P_α) para o próximo apenas o vetor do lado direito sofre alteração (nas últimas posições, associadas ao parâmetro α). Assim, a técnica de pós-otimização, aplicando o algoritmo Simplex dual, pode acelerar ainda mais a execução do Algoritmo 1.

Exemplo de funcionamento. Considere o conjunto de fluxos $F = \{f_1, f_2, f_3, f_4, f_5\}$ a ser roteado no grafo da Figura 1(a), onde o nó 2 é a origem de todos os fluxos, o nó 3 é o destino dos fluxos f_1, f_2 e f_3, e o nó 4 é o destino de f_4 e f_5.



(a) Grafo original. (b) Primeira iteração.

(c) Segunda iteração. (d) Terceira iteração.

Figura 1: Exemplo de execução do Algoritmo 1.

Na primeira iteração tem-se $\alpha = \infty$ e obtemos a solução ótima x^1, com $z_1(x^1) = 4$ (quatro fluxos na aresta $(2,3)$) e $z_2(x^1) = 7$ (sete saltos para o roteamento dos fluxos), conforme Figura 1(b). Na segunda iteração, resolvendo (P_α) para $\alpha = 3$, obtemos a solução ótima x^2, com $z_1(x^2) = 3$ e $z_2(x^2) = 7$, conforme Figura 1(c). Como ocorreu empate no valor de z_2, eliminamos x^1 do conjunto de soluções. Veja que, de fato, x^2 domina x^1. Na terceira iteração, para $\alpha = 2$, obtemos x^3 com $z_1(x^3) = 2$ e $z_2(x^3) = 9$, conforme Figura 1(d). Na iteração seguinte o algoritmo é finalizado, após detectar que (P_α), $\alpha = 1$, é inviável. A tabela abaixo resume a execução completa do algoritmo.

Tabela 1: Soluções geradas pelo Algoritmo 1.

Iteração (k)	α	(P_α)	$z_1(x^k)$	$z_2(x^k)$	X^*
1	∞	Viável	4	7	$\{x^1\}$
2	3	Viável	3	7	$\{x^2\}$
3	2	Viável	2	9	$\{x^2, x^3\}$
4	1	Inviável	-	-	$\{x^2, x^3\}$

A seguir apresentamos resultados que comprovam a otimalidade do Algoritmo 1.

Proposição 1. *No final do Algoritmo 1 todas as soluções em X^* são Pareto-ótimas para (P).*

Justificativa. Seja x uma solução Pareto-ótima qualquer de (P) e x^k uma solução qualquer em X^*, no final do algoritmo, obtida resolvendo o problema (P_α). Mostraremos, a seguir, que x^k não é dominado por x, o que implica que x^k é Pareto-ótima de (P). Para o caso em que $z_1(x) > z_1(x^k)$ temos, obviamente, que x^k não é dominado por x. No caso de $z_1(x) \leq z_1(x^k)$, temos que x é viável para (P_α). Como a solução ótima em relação a z_2, obtida em (P_α), foi x^k, temos $z_2(x^k) \leq z_2(x)$. Logo, x^k somente seria dominado por x se $z_2(x) = z_2(x^k)$ para $z_1(x) < z_1(x^k)$. Mas, $z_1(x) < z_1(x^k)$ implica em x ser viável para algum $(P_{\bar{\alpha}})$, $\bar{\alpha} < \alpha$. Neste caso a comparação feita na iteração seguinte a que x^k foi obtida nos levaria a excluir x^k, pois $z_2(x) = z_2(x^k)$, o que seria um absurdo, uma vez que

$x^k \in X^*$, no final do algoritmo. Observe que se x^k foi obtido na última iteração do algoritmo, então z_1 atinge seu valor mínimo em x^k, consequentemente, $z_1(x^k) \leq z_1(x)$. No caso de $z_1(x^k) = z_1(x)$ teríamos x viável em (P_α) o que levaria $z_2(x^k) \leq z_2(x)$. Portanto, x^k não é dominado por x. □

Proposição 2. *Para qualquer solução x, Pareto-ótima para (P), existe uma solução equivalente em X^* no final do Algoritmo 1.*

Justificativa. Seja α o menor valor inteiro maior ou igual a $z_1(x)$ para o qual uma iteração do algoritmo foi executada. Isto implica que x é viável para (P_α). Seja x^k a solução ótima de (P_α) obtida nesta iteração. Consequentemente, temos que $z_2(x^k) \leq z_2(x)$. Por outro lado, como x é solução Pareto-ótima, então $z_1(x^k) \geq z_1(x)$. Veja que o caso $z_1(x^k) > z_1(x)$ é impossível, pois com isso teríamos executado uma outra iteração $\bar{\alpha} < \alpha$ tal que $\bar{\alpha} \geq z_1(x)$. Logo, $z_1(x^k) = z_1(x)$ e $z_2(x^k) = z_2(x)$. □

Portanto, o conjunto X^*, no final do Algoritmo 1, é um conjunto mínimo completo de soluções Pareto-ótimas para (P). Todas as soluções $x^k \in X^*$ são Pareto-ótimas (Teorema 1) e para qualquer solução Pareto-ótima x existe uma solução equivalente em X^*, isto é, com o mesmo vetor objetivo (Teorema 2). Além disso, pelo Passo 6 do Algoritmo 1, todas as soluções $x \in X^*$ têm vetores objetivos distintos.

Proposição 3. *A complexidade do Algoritmo 1 é $O(r\,g(\alpha))$, onde r é o número de fluxos e $g(\alpha)$ é a complexidade de (P_α).*

Justificativa. O pior caso, na primeira iteração, é que exista pelo menos uma aresta que por ela passe todos os r fluxos, isto é, o gargalo é igual a r. Na segunda iteração, o pior caso, seria que o gargalo fosse igual a $r-1$. Seguindo desta forma, teríamos exatamente r iterações. Portanto, como a complexidade de cada iteração é $g(\alpha)$, temos que a complexidade do Algoritmo 1 é $O(r\,g(\alpha))$. □

Consequência da Proposição 1. *A cardinalidade de X^*, no final do Algoritmo 1, é limitada superiormente pelo número de fluxos, r.*

Justificativa. Pelo Teorema 3 , o número máximo de iterações é r. Além disso, pelo Algoritmo 1, no máximo uma solução é inserida em X^*, por iteração. Logo, a cardinalidade de X^* é limitado superiormente pelo número de fluxos. □

4 Resultados Computacionais

Os resultados computacionais, aqui apresentados, são referentes à implementação do Algoritmo 1 para a resolução do problema (P), definido sobre grafos aleatórios. As implementações foram feitas em linguagem C++ e compiladas com o Microsoft Visual C++ 2010. O modelo matemático foi implementado em C++ utilizando a biblioteca Concert do CPLEX 12.4 e os experimentos foram executados em um PC com processador Intel Core 2 Duo T6400 e memória RAM de 4Gb.

Grafos com $n = 50$, 100 e 150 nós foram obtidos gerando aleatoriamente $\widehat{m}.n$ arestas, onde cada nó possui grau de entrada e saída iguais a $\widehat{m}$. A densidade do grafo deu origem a dois grupos de instâncias, um com $\widehat{m} = 4$ e outro com $\widehat{m} = 8$. Os nós de origem e de destino de cada fluxo $f \in F$ também foram gerados aleatoriamente, considerando três configurações distintas. Na primeira, todos os fluxos têm o mesmo destino e as origens são variadas. Na segunda, todos os fluxos têm

a mesma origem e o mesmo destino. E, na terceira configuração, a origem e o destino são variados. Não apresentamos o caso em que os fluxos têm a mesma origem e destinos variados, pois os resultados são análogos aos da primeira configuração. A quantidade de fluxos considerada foi de $r = 25, 50, 75, 100, 150, 200$ e 300.

As Tabelas 2 e 3 mostram o número de nós (n), a quantidade de fluxos (r), a cardinalidade do conjunto mínimo completo ($|X^*|$), o número de iterações em que P_α foi resolvido (It) e o tempo de CPU, em segundos, para executar o algoritmo (t). Os resultados apresentados são valores médios para 10 diferentes instâncias do conjunto de dados, com $\widehat{m} = 4$ e $\widehat{m} = 8$, respectivamente.

Tabela 2: Resultados para $\widehat{m} = 4$.

Tipos de Fluxos	n	r	$\lvert X^*\rvert$	It	t
Origens	50	25	1	5	4,76
quaisquer e	50	50	1	8	11,24
mesmo destino	50	150	1	10	32,97
Mesma	50	25	14	16	22,31
origem e	50	50	21	26	50,60
mesmo destino	50	150	61	69	265,15
Origens	50	25	2	3	4,17
quaisquer e	50	50	2	4	26,91
destinos quaisquer	50	150	3	7	36,07
Origens	100	50	1	7	14,89
quaisquer e	100	100	1	10	37,55
mesmo destino	100	300	1	20	214,85
Mesma	100	50	13	18	48,14
origem e	100	100	16	23	83,14
mesmo destino	100	300	136	143	1573,47
Origens	100	50	2	3	10,87
quaisquer e	100	100	2	5	30,65
destinos quaisquer	100	300	3	8	167,82
Origens	150	75	1	7	34,19
quaisquer e	150	150	1	11	100,75
mesmo destino	150	300	1	20	305,92
Mesma	150	75	12	18	69,44
origem e	150	150	46	53	397,56
mesmo destino	150	300	24	37	550,47
Origens	150	75	2	4	20,42
quaisquer e	150	150	2	5	72,82
destinos quaisquer	150	200	2	6	92,35

Tabela 3: Resultados para $\widehat{m} = 8$.

Tipos de Fluxos	n	r	$\lvert X^*\rvert$	It	t
Origens	50	25	2	5	5,61
quaisquer e	50	50	1	6	12,62
mesmo destino	50	150	2	10	54,51
Mesma	50	25	13	19	24,54
origem e	50	50	33	37	80,93
mesmo destino	50	150	90	95	567,00
Origens	50	25	1	2	3,29
quaisquer e	50	50	2	3	7,83
destinos quaisquer	50	150	2	4	44,70
Origens	100	50	1	5	15,44
quaisquer e	100	100	1	8	49,18
mesmo destino	100	300	1	14	271,79
Mesma	100	50	25	34	98,83
origem e	100	100	45	56	355,5
mesmo destino	100	300	124	143	2864,76
Origens	100	50	2	2	7,32
quaisquer e	100	100	2	3	24,53
destinos quaisquer	100	300	2	5	151,89
Origens	150	75	2	8	50,81
quaisquer e	150	150	1	9	110,02
mesmo destino	150	300	1	15	414
Mesma	150	75	47	53	348,76
origem e	150	150	97	101	1387,04
mesmo destino	150	300	187	193	5125,61
Origens	150	75	2	3	18,24
quaisquer e	150	150	2	3	52,83
destinos quaisquer	150	200	2	4	87,74

A Tabela 4 apresenta a porcentagem da quantidade de soluções geradas pelo Algoritmo 1 que são Pareto-ótimas (Rel_1) e a porcentagem da quantidade de iterações em relação ao limite máximo que poderia ocorrer (Rel_2), conforme o Corolário 1. Esses resultados são valores médios para 60 diferentes instâncias de conjunto de dados, sendo dez de cada um dos seis cenários, considerando $r < n$, $r = n$ e $r > n$.

Tabela 4: Resultados médios para $\widehat{m} = 4$ e $\widehat{m} = 8$.

		$r < n$		$r = n$		$r > n$	
Tipos de Fluxos	**Relação**	**Média**	**Desvio Padrão**	**Média**	**Desvio Padrão**	**Média**	**Desvio Padrão**
Origens quaisquer e	Rel_1	29%	18%	17%	7%	11%	11%
mesmo destino	Rel_2	13%	5%	9%	3%	6%	2%
Mesma origem e	Rel_1	64%	4%	65%	4%	65%	4%
mesmo destino	Rel_2	53%	3%	47%	3%	97%	15%
Origens quaisquer e	Rel_1	72%	22%	60%	21%	54%	18%
destinos quaisquer	Rel_2	6%	3%	4%	1%	3%	2%

A seguir são comentados os resultados computacionais, com base nos valores apresentados

nas Tabelas 2, 3 e 4.

a) Todos os fluxos com origens quaisquer e mesmo destino

Para este cenário é fácil ver que o gargalo mínimo (b_{min}) é dado por $b_{min} = \left\lceil \frac{r}{\widehat{m}} \right\rceil$, pois todos os fluxos têm o mesmo destino. Pelo fato dos fluxos terem origens distintas e pela estrutura k-regular do grafo, mesmo reduzindo o limite na restrição do gargalo, continuamos encontrando, em muitos casos, caminhos com a mesma quantidade de saltos para os fluxos. Isto justifica o baixo número de soluções Pareto-ótimas em X^*.

O aumento no número de fluxos implicou no aumento do número de iterações, porém, a proporção de iterações em relação ao limite máximo de iterações possíveis foi reduzida. Já a quantidade média de soluções Pareto-ótimas não sofreu alterações significativas. Além disso, para todas as instâncias deste grupo, o número de iterações foi, no máximo, 13% do limite superior possível de iterações.

b) Todos os fluxos com a mesma origem e mesmo destino

Como o gargalo está entre $b_{min} = \left\lceil \frac{r}{\widehat{m}} \right\rceil$ e $b_{max} = r$, pois no pior caso podemos ter todos os r fluxos passando por uma certa aresta, então podemos ter até $(r - \left\lceil \frac{r}{\widehat{m}} \right\rceil + 1)$ iterações. Observe que para esta classe de instâncias, por termos a mesma origem e o mesmo destino para todos os fluxos, é muito provável termos todos os fluxos passando pelo mesmo caminho. Neste caso, a primeira solução atinge o gargalo máximo, que é $b_{max} = r$. Em muitos casos, para este grupo, ocorreu de obter, em cada iteração, uma solução com uma piora no número de saltos e melhorando o gargalo em poucas unidades. Isto é o que justifica o alto número de soluções Pareto-ótimas.

c) Todos os fluxos com origens e destinos quaisquer

Como para estas instâncias as origens e os destinos dos fluxos são distintos, tivemos um gargalo muito pequeno já na primeira iteração. Neste caso o número de iterações e o número de soluções Pareto-ótimas são extremamente baixos, como pode ser observado nas tabelas. De fato, o número de iterações para estas instâncias foi sempre menor ou igual a 6% do limite máximo de iterações possíveis.

5 Considerações Finais

Os algoritmos apresentados em [Berman et al., 1990; Pinto et al., 2009; Pinto e Pascoal, 2010; Bornstein et al., 2012] são eficientes (polinomiais), mas limitados a problemas em que os pesos dos elementos que compõem a solução sejam fixos.

Problemas em redes similares ao apresentado aqui, isto é, que também visam rotear um conjunto de fluxos por caminhos minimizando o gargalo da rede e o comprimento dos caminhos, são tratados com algoritmos heurísticos em [Gálvez e Ruiz, 2013; De Melo et al., 2014]. Uma diferença na formulação dos problemas é que, enquanto nestes dois trabalhos o comprimento de cada caminho é considerado individualmente, aqui elaboramos uma única função totalizadora que busca minimizar o número de saltos para todos os caminhos. Isto nos possibilitou tratar o problema de forma exata, o que consiste na contribuição deste trabalho.

Nos resultados computacionais, observamos que para instâncias com fluxos de origens e/ou destinos variados, o número de soluções Pareto-ótimas no conjunto mínimo completo é extremamente baixo. Isto consiste em um resultado importante, pois facilita a tomada de decisão em torno de qual solução escolher para fazer o roteamento dos fluxos.

Pesquisas futuras podem lidar com extensões das ideias apresentadas aqui. Acrescentar restrições ao modelo de modo a impor um limite superior para o comprimento de cada fluxo; considerar capacidades para as arestas; lidar com fluxos de pesos variados; e, considerar outros fluxos já existentes na rede.

Agradecimento. Os autores agradecem os valiosos comentários e sugestões feitos pelos avaliadores deste trabalho.

Referências

Ahuja, R. K., Magnanti, T. L. e Orlin, J. B. (1993). *Network Flows: Theory, Algorithms and Applications*. 1st Edition. Prentice-Hall.

Alves, M. J. e Costa, J. P. (2012). *Programação Linear Inteira e Inteira-Mista Multiobjetivo: Conceitos Fundamentais e Métodos*. Congreso latino-liberoamericano de Investigación Operativa e Simpósio Brasileiro de Pesquisa Operacional.

Berman, O., Einav, D. e Handler, G. (1990). *The constrained Bottleneck Problem in Networks*. Operations Research 38 (1): 178-181.

Bornstein, C. T., Maculan, N., Pascoal, M. e Pinto, L. L. (2012). *Multiobjective combinatorial optimization problems with a cost and several bottleneck objective functions: An algorithm with reoptimization*. Computers & Operations Research, 39: 1969-1976.

Clímaco, J. C. N. e Martins, E.Q.V. (1982). *A bicriterion shortest path algorithm*. European Journal of Operational Research, 11: 399-404.

Clímaco, J. C. N, Ferreira, C. e Captivo, M. E. (1997). *Multicriteria integer programming: an overview of the different algorithmic approaches*. Multicriteria Analysis, Springer-Verlag, Berlin, 248-258.

De Mello, M. O. M. C., Borges, V. C. M., Pinto, L. L. e Cardoso, K. V. (2014). *Load balancing routing for path length and overhead controlling in Wireless Mesh Networks*. In Computers and Communication (ISCC), 2014 IEEE Symposium on (pp. 1-6). IEEE.

Ehrgott, M. e Gandibleux, X. (2002). *Multi Objective Combinatorial Optimization, in: M. Ehrgott, X. Gandibleux (Eds.), Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*. Vol. 52, Kluwer's International Series in Operations Research and Management Science.

Ehrgott, M. (2006). *A discussion of scalarization techniques for multiple objective integer programming*. Annals of Operations Research, 147(1), 343-360.

Gálvez, J. J. e Ruiz, P. M. (2013). *Efficient Rate Allocation, Routing and Channel Assignment in Wireless Mesh Networks Suportting Dynamic Traffic Flows*. Ad Hoc Networks, vol. 11 (6): 1765 - 1781.

Hansen, P. (1980). *Bicriterion path problems*. In: Multiple Criteria Decision Making: Theory and Applications, Lectures Notes in Economics and Mathematical Systems, 177, G. Fandel and T. Gal (eds), Springer, Heidelberg, 109-127.

Klein, D., Hannan, E. (1982). *An algorithm for the multiple objective integer linear programming problem*. European Journal of Operational Research, 9(4), 378-385.

Martins, E.Q.V. (1984). *On a special class of bicriterion path problems*. European Journal of Operational Research 17: 85-94.

Pinto, L. L., Bornstein, C. T. e Maculan, N. (2009). *The criterion shorthest path problem with at least two bottleneck objective functions*. European Journal of Operational Research; 198: 387-91.

Pinto, L. L. e Pascoal, M. M. B. (2010). *On algorithms for the tricriteria shortest path problem with two bottleneck objective functions*. Computers & Research, 37: 1774-1779.

Ulungu, E. L. e Teghem, J. (1994). *Multi-objective combinatorial optimization problems: A survey*. Journal of Multi-Criteria Decision Analysis, 3 (2): 83-104.