Um Estudo Experimental do Problema de Caminhos Mínimos Multiobjetivo

Wagner Schmitt
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
wschmitt@inf.ufrgs.br

Leonardo T. C. Bezerra
DIMAp – Universidade Federal do Rio Grande do Norte(UFRN)
leo.adoro@gmail.com

Luciana S. Buriol
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
buriol@inf.ufrgs.br

Elizabeth F. G. Goldbarg
DIMAp – Universidade Federal do Rio Grande do Norte(UFRN)
beth@dimap.ufrn.br

Marco Goldbarg
DIMAp – Universidade Federal do Rio Grande do Norte(UFRN)
marcocgold@gmail.com

Marcus Ritt[1]
[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
marcus.ritt@inf.ufrgs.br

RESUMO

O problema de caminhos mínimos de fonte única é um problema com solução polinomial bem conhecido na literatura. No entanto, quando cada arco possui dois ou mais valores associados, o problema se torna NP-difícil e, apesar de modelar diversas situações reais, ainda é pouco explorado. Neste artigo, apresentamos um estudo experimental sobre algoritmos exatos para resolução do problema de caminhos mínimos multiobjetivo. Mais especificamente, comparamos resultados de algoritmos de correção de rótulos utilizando sete classes diferentes de redes. Os resultados são avaliados em função do tempo de execução e do número de soluções não-dominadas.
PALAVRAS CHAVE. Caminho mínimo multiobjetivo, Algoritmo de correção de rótulos.
Área Principal: Teoria e Algoritmos em Grafos

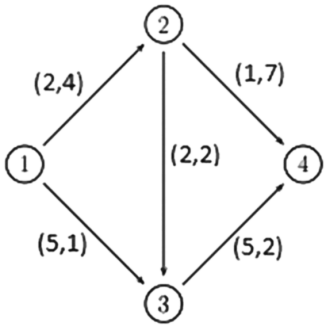
ABSTRACT

The single-source shortest path is a problem with polynomial solution well known in the literature. Nevertheless, when two or more values are associated to each edge, the problem becomes NP-hard and, although it models several real applications, it is still unexplored. In this paper, we present an experimental study on exact algorithms to solve the multi-objective shortest path problem. More specifically, we compare the results of label correction algorithms using seven different classes of networks. The results are evaluated according to the execution time and the number of non-dominated solutions.
KEYWORDS. Multi-objective shortest path, Label correction algorithm.
Main Area: Theory and Graph Algorithms

1 Introdução

Problemas de caminhos mínimos de fonte única têm sido amplamente estudados na literatura [Goldberg (2007), Demetrescu (2006), Likhachev (2008)], visto que existe uma grande área de aplicações reais que necessitam encontrar soluções ótimas, como nas áreas de telecomunicações e transportes [Current (1993)]. Contudo, na modelagem de problemas reais, muitas vezes é necessário extendê-los para mais de um objetivo [Vincke (1974)], podendo-se considerar distância, tempo, risco, sobrecarga, entre outros.

Neste estudo abordamos o problema de caminhos mínimos de fonte única biobjetivo (BSP - do inglês *Biojective Shortest Path Problem*) e com um número arbitrário de objetivos (MSP - do inglês *Multiobjective Shortest Path problem*). Em ambos os casos consideramos que os pesos dos arcos são sempre valores não negativos. Em problemas com apenas um objetivo o melhor caminho é sempre o caminho ótimo e este algoritmo é resolvido em tempo $O(m + n \log n)$ pelo algoritmo $Dijkstra$ usando heaps de Fibonacci. Já nos problemas multiobjetivos, dificilmente encontraremos um caminho que apresente os menores valores considerando todos os objetivos, visto que podem ocorrer conflitos entre eles. Portanto, o critério de otimalidade é substituído por Pareto otimalidade. Em um problema multiobjetivo podemos ter muitas soluções Pareto ótimas, também chamadas de soluções eficientes (ver Definição 1). De forma simples, podemos dizer que uma solução é eficiente ou Pareto ótima se não for possível encontrar outra solução que seja melhor do que a atual em pelo menos um objetivo sem simultaneamente piorar pelo menos algum outro objetivo. Utilizando o grafo da Figura 1, podemos ilustrar de forma mais clara o conceito de soluções eficientes; neste grafo cada aresta possui dois objetivos e queremos encontrar os caminhos Pareto ótimos do vértice inicial 1 até o vértice final 4. Assim, verificamos que há três soluções eficientes (os caminhos A, B e C). Todas as três pertencem ao conjunto de soluções eficientes, pois o caminho A tem o melhor custo $c1$, C tem o melhor custo $c2$ e B possui $c1$ melhor do que C e $c2$ melhor do que A, então dizemos A, B e C possuem critérios não-dominantes entre si.



Caminhos	Custos (c1,c2)
A: 1-2-4	(3,11)
B: 1-2-3-4	(9,8)
C: 1-3-4	(10,3)

Figura 1. Grafo com três soluções Pareto ótimas, descritas pelos caminhos A, B e C.

[Hansen (1980)] provou que o número de soluções eficientes para problemas biobjetivo pode crescer exponencialmente com o número de nodos. Além disso, [Serafini (1986)] provou que o problema de caminhos mínimos para mais de um objetivo pertence a classe de problemas NP-Difícil, por redução do problema da mochila binária. Contudo, em aplicações práticas, como grafos rodoviários, muitas vezes tem-se um número pequeno de soluções eficientes [Raith (2009)], principalmente em problemas biobjetivo. Portanto, é possível encontrarmos todas as soluções eficientes em um tempo relativamente baixo.

Este artigo tem o propósito de estudar diferentes classes de grafos no contexto multiobjetivo, com o intuito de avaliar os resultados em função do tempo e do número de soluções. O artigo está organizado da seguinte forma: nas Subseções 1.1 e Subseção 1.2 apresentamos formalmente o problema MSP e uma revisão bibliográfica, respectivamente. Logo após, na Seção 2, apresentamos o algoritmo de correção de rótulos proposto em [Brumbaugh-Smith (1989)]. Na Seção 3 apresentamos uma proposta de generalização do algoritmo de correção de rótulos biobjetivo para solucionar problemas com mais de dois objetivos. Resultados computacionais são apresentados e discutidos na Seção 4. Finalmente, na Seção 5 apresentamos as conclusões deste trabalho.

1.1 Problema MSP

Seja um grafo orientado $G = (N, A)$, com $|N| = n$ nodos e $|A| = m$ arcos. A cada arco $(i, j) \in A$ estão associados p custos positivos $C_{ij} \in \mathbb{N}^p$. O objetivo é encontrar o conjunto de caminhos eficientes do nó inicial $s \in N$, até o nodo final $t \in N$. O problema MSP é tradicionalmente formulado da seguinte forma [Martins (1984)]:

$$\min Z(x) = \begin{cases} z_1(x) = \sum\limits_{(i,j)\in A} c_{ij}^1 \cdot x_{ij}, \\ \cdots, \\ z_p(x) = \sum\limits_{(i,j)\in A} c_{ij}^p \cdot x_{ij}, \end{cases} \tag{1}$$

$$\text{sujeito a} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{se } i = s, \\ 0 & \text{se } i \neq s, t, \\ -1 & \text{se } i = t, \end{cases} \tag{2}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A \tag{3}$$

Essa formulação tem a mesma forma de um problema de fluxo em redes, portanto podemos ver x como um vetor de fluxos sobre os arcos [Raith (2009)]. Porém nesta formulação x_{ij} assume valores binários, indicando se o arco (i, j) faz parte de uma solução eficiente.

O conjunto viável X, contendo todas soluções factíveis do problema, é descrito pelas restrições (2) e (3).

Definição 1: Uma solução $x' \in X$ é eficiente se, e somente se, não existe $x \in X$ tal que $z_k(x) \leq z_k(x')$ para $k = 1, \ldots, p$ e $z_i(x) < z_i(x')$ para algum $i \in \{1, \ldots, k\}$.

1.2 Literatura Relacionada

Esta seção revisa propostas de métodos exatos para resolução de problemas BSP e MSP. Neste escopo, há duas abordagens exatas encontradas na literatura para resolução deste problema [Skriver (2000)a]: rotulação de nodos e de caminho/árvore. O primeiro tem como submétodos correção de rótulos e seleção de rótulos, enquanto que o segundo possui como submétodos o método de 2-fases e o K-ésimo menor caminho.

Abordagem	Referência
K-ésimo menor caminho	BSP - Clímaco e Martins [Climaco (1982)]
Método de 2 fases	BSP - Mote et. al. [Mote (1991)]
Seleção de Rótulos	BSP- Hansen [Hansen (1980)] MSP - Tung e Chow [Tung (1992)] MSP - Martins [Martins (1984)]
Correção de Rótulos	MSP - Corley e Moon [Corley (1985)] BSP - Brumbaugh-Smith[Brumbaugh-Smith (1989)] MSP - Martins e Dos Santos [Martins EQV (1999)] BSP - Skriver e Andersen [Skriver (2000)b]

Tabela 1. Literatura de soluções exatas para o problema BSP e MSP.

A Tabela 1 apresenta um resumo das abordagens encontradas na literatura. A abordagem do K-ésimo menor caminho de [Climaco (1982)] utiliza uma busca ordenada iniciando com a minimização do primeiro critério e o menor valor possível encontrado no segundo. O primeiro critério é gradualmente relaxado, encontrando o melhor caminho com respeito ao segundo critério. Este algoritmo requer a enumeração de todos os caminhos em relação a ordem dos custos, o que é bastante custoso. Contudo, variações desta abordagem já foram propostas, como o método de *próximo menor caminho* [Carlyle (2005)] que, segundo os autores, é o algoritmo mais rápido dedicado a resolver o problema do K-ésimo menor caminho. Porém, este algoritmo é rápido apenas em alguns tipos específicos de instâncias, e é bastante demorado em outras, principalmente em problemas onde as soluções dominadas do conjunto viável são muito próximas às soluções não-dominadas. Dessa forma, muitos caminhos viáveis são enumerados e precisam ser analisados, como acontece em instâncias com estrutura de grade [Raith (2009)].

Já o método de duas fases proposto por [Mote (1991)] utiliza uma abordagem diferente. Na primeira fase do algoritmo são encontradas todas as soluções eficientes suportadas. Essas soluções têm a vantagem de serem obtidas por meio do método da soma dos pesos [Raith (2009)]. Na segunda fase, são encontradas todas as soluções eficientes não-suportadas. Para isso, utiliza-se qualquer outro método, como o enumerativo ou de correção de rótulos, porém o espaço de busca agora é bastante restrito devido as informações obtidas na primeira fase. [Raith (2009)] realizou diversos experimentos com diversas abordagens diferentes para primeira e segunda fase desse algoritmo, e os autores constataram que é um método bastante competitivo para resolução de problemas BSP.

O método de rotulação de nodos segue um caminho diferente das abordagens de caminho/árvore. São métodos estendidos diretamente da versão com apenas um objetivo. A abordagem de seleção de rótulos de [Martins (1984)] pode ser vista como uma generalização do algoritmo de Dijkstra para o caso multiobjetivo.

O método de correção de rótulos é um dos mais eficientes conhecidos para resolução de problemas BSP [Raith (2009), Skriver (2000)b]. Como este algoritmo foi proposto para o caso biobjetivo, apresentamos na Seção 3 uma generalização desse algoritmo para resolução de problemas multiobjetivos. A próxima seção apresenta este método de forma mais detalhada.

2 Algoritmo de Correção de Rótulos Biobjetivo (CRB)

O algoritmo de correção de rótulos para o problema BSP, apresentado no O Algoritmo 1, proposto por [Skriver (2000)b] é uma extensão direta da versão para apenas um objetivo. A diferença é que um nodo pode ter muitos rótulos, sendo que cada um representa o custo de um caminho diferente, e os rótulos de cada nodo não são dominantes entre si.

O algoritmo recebe como entrada o grafo G biobjetivo, um vetor bidimensional com os custos c_1 e c_2 de cada arco, e um nó inicial s. Além disso, $D(i)$ representa um conjunto contendo os rótulos de um nó i, e cada rótudo $l = (v_1, v_2) \in D(i)$, indica o custo v_1 e v_2, referente aos objetivos um e dois de G, respectivamente, de percorrer um caminho do nó inicial s ao nó i. Inicialmente, apenas o nó s recebe um rótulo $(0, 0)$, e é adicionado à lista L que contém os nós a serem analisados. Todos os rótulos em um nó particular i são extendidos para todos os seus vizinhos, retirando-se os rótulos dominados de cada conjunto; isso é feito pelo procedimento *merge*. Cada vez que o conjunto de rótulos de um nó é atualizado, este é adicionado à L para ser verificado posteriormente. Nós marcados para verificação são analisado em ordem FIFO, pois, segundo [Brumbaugh-Smith (1989)], este método de verificação é o que obteve melhor desempenho. Além disso, implementamos também a verificação de pré-dominância do conjunto inteiro de rótulos proposta por [Skriver (2000)b], onde podemos verificar a dominância de um conjunto inteiro de rótulos sobre outro de forma rápida, apenas comparando o primeiro e último rótulo de cada conjunto, dado que os rótulos estão ordenados de forma lexicográfica em cada um dos conjuntos.

Algoritmo 1 - Correção de Rótulos Biobjetivo (CRB)

1: **input:** $G(N, A), c = (c_1, c_2), s$
2: $L \leftarrow s$ ▷ Lista com os nodos a serem analisados em ordem FIFO
3: $D(s) \leftarrow \{(0, 0)\}$ e $D(i) \leftarrow \emptyset, i \in N \setminus \{s\}$ ▷ $D(i)$ Conjunto de rótulos de um nodo i
4: **while** $L \neq \emptyset$ **do**
5: $L \leftarrow L - \{i\}$ ▷ retira o primeiro nodo da lista, ordem FIFO
6: **for all** (i, j) com $j \in \{k \in N | (i, k) \in A\}$ **do**
7: **if** $D(i) + c_{ij}$ é dominado por $D(j)$ **then**
8: descarta aresta ▷ D(j) não é modificado
9: **else**
10: $D(j) \leftarrow$ merge$(D(i) + c_{ij}, D(j))$ ▷ soma aos rótulos de $D(i)$ o custo de (i, j)
11: **end if**
12: **if** $D(j)$ foi modificado e $j \notin L$ **then**
13: $L \leftarrow L + \{j\}$
14: **end if**
15: **end for**
16: **end while**
17: **output:** $D(i)$ ▷ $D(i)$ contém o custo de todos caminhos eficientes do nodo s ao nodo i

O procedimento *merge* é o componente computacionalmente mais caro do algoritmo. [Brumbaugh-Smith (1989)] apresentaram um *merge* linear nos tamanhos M e N dos conjuntos em questão, cujo pseudo-código se encontra no Algoritmo 2. Porém, o *merge* proposto apenas funciona para problemas com exatamente dois objetivos. Esse método necessita que os rótulos de cada conjunto estejam ordenados de forma crescente pelo primeiro objetivo. Dessa forma, automaticamente eles estarão também ordenados de forma decrescente pelo segundo objetivo. Isso acontece porque um rótulo r_1 possui o primeiro objetivo maior do que um outro rótulos r_2, ele obrigatoriamente terá o segundo objetivo menor, caso contrário r_2 seria dominado pelo rótulo r_1 e, portanto, removido do conjunto. Dessa forma, com os conjuntos ordenados, com apenas uma comparação (com o último rótulo do conjunto) podemos verificar se um rótulo está dominado.

Procedimento 2 - Merge biobjetivo

input: D_1, D_2 ▷ D_1 e D_2 são dois conjuntos ordenados de rótulos
$D_M \leftarrow \emptyset$ ▷ conjunto $D_1 \cup D_2 \backslash$(rotulos dominados de $D_1 \cup D_2$)
while $D_1 \neq \emptyset$ e $D_2 \neq \emptyset$ **do**
  if comparação lexicográfica entre $first(D_1)$ e $first(D_2)$ **then**
    if $D_M = \emptyset$ ou $first(D_1)$ não é dominado por $last(D_M)$ **then**
      $D_M \leftarrow D_M + \{first(D_1)\}$
    end if
    $D_1 \leftarrow D_1 - \{first(D_1)\}$ ▷ remove elemento no início de D_1
  else
    if $D_M = \emptyset$ ou $first(D_2)$ não é dominado por $last(D_M)$ **then**
      $D_M \leftarrow D_M + \{first(D_2)\}$
    end if
    $D_2 \leftarrow D_2 - \{first(D_2)\}$ ▷ remove elemento no início de D_2
  end if
  if $D_1 = \emptyset$ **then**
    $D_M \leftarrow D_M \cup (D_2 \backslash$(rótulos não dominados por $last(D_M)))$
    $D_2 \leftarrow \emptyset$
  else if $D_2 = \emptyset$ **then**
    $D_M \leftarrow D_M \cup (D_1 \backslash$(rótulos não dominados por $last(D_M)))$
    $D_1 \leftarrow \emptyset$
  end if
end while
output: $D_M = D_1 \cup D_2 \backslash$(rotulos dominados de $D_1 \cup D_2$)

3 Generalização do Algoritmo de Correção de Rótulos para um Número Arbitrário de Objetivos (CRM)

Esta seção tem o objetivo de apresentar a generalização do algoritmo da seção anterior para o caso multiobjetivo, ou seja, considerando um número arbitrário de objetivos. A generalização em muito se assemelha ao algoritmo original. No entanto, as otimizações propostas não se aplicam ao caso multiobjetivo, e o algoritmo *merge* deve ser adaptado. Quando os rótulos têm mais de dois objetivos, o *merge* deve ser projetado de forma a comparar um rótulo contra todos os outros do conjunto para verificar se ele está dominado. Além disso, também não é mais possível fazer a verificação de pré-dominância do conjunto, visto que *merge* biobjetivo também se valia dessa mesma propriedade de ordenação. O Algoritmo 3 descreve o procedimento *merge* multiobjetivo.

Procedimento 3 - Merge multiobjetivo

input: D_1, D_2 ▷ D_1 e D_2 são dois conjuntos de de rótulos ordenados lexicograficamente
$D_u \leftarrow$ ordena$(D_1 \cup D_2)$ ▷ ordena D_u lexicograficamente
$S \leftarrow first(D_u)$ ▷ conjunto contendo os rótulos não dominados
for all $r_i \in D_u \backslash first(D_u)$ **do**
  if r_i não é dominado por nenhum rótulo de S **then**
    $S \leftarrow S + \{r_i\}$
  end if
  if r_i domina alguma solução $s \in S$ **then**
    $S \leftarrow S - \{s\}$
  end if
end for
output: $S = D_1 \cup D_2 \backslash$(rotulos dominados de $D_1 \cup D_2$)

No *merge* multiobjetivo proposto, a complexidade no pior caso é $O(|N| \times |M|)$, onde $|N|$ e $|M|$ são as dimensões dos conjuntos D_1 e D_2. Isso pode ser visto quando nenhum rótulo r_i é dominado. Por outro lado, quando todos os rótulos inseridos no conjunto S são dominados, ele apresenta uma complexidade linear $O(|M| + |N|)$, pois o custo para ordenar os conjuntos D_1 e D_2 é linear, visto que cada conjunto já está ordenado. Além disso, cada nodo irá ser testado contra apenas um, o rótulo inicial do conjunto S, caracterizando o melhor caso do algoritmo.

4 Resultados Computacionais

Nesta seção apresentamos e discutimos os resultados experimentais comparando o algoritmo biobjetivo (CRB) correspondente ao Algorithmo 1, com a nossa versão generalizada (CRM) em relação ao tempo e ao número de soluções eficientes. Todos os experimentos foram realizados em um processador Intel Core I7 $2,8$MHz, com 12GB de RAM, rodando na plataforma Linux Ubuntu 10.1. Os algoritmos foram implementados em C++ e compilados com g++ versão 4.3.

As instâncias usadas são em parte de origem prática, enquanto outras são sintéticas. A primeira classe G_1 representa redes de tráfegos rodoviários de cidades dos EUA obtidas em [Gera (2001)]. As quatro instâncias criadas são todas aquelas que possuem coordenadas dos nós disponíveis. Utilizamos como objetivos o tempo associado a cada arco (objetivo-1) e a distância euclidiana (objetivo-2) obtida através dos valores das coordenadas dos nodos. Além disso, adicionamos mais dois custos para cada arco: um valor aleatório entre $[1, 4000]$ (objetivo-3) e um valor constante igual a um (objetivo-4), de forma que o caminho mínimo considerando este último objetivo seria aquele com o menor número de arestas (*hop count*).

As instâncias das outras seis classes $G_2, ..., G_7$ foram geradas a partir do gerador de instâncias de caminho mínimo SPLIB [Cherkassky (1994)]. As características das instâncias de cada classe podem ser vistas na Tabela 2. $G2$ possui redes sem ciclos. Instâncias das classes $G3$ são construídas criando-se inicialmente um ciclo hamiltoniano, e cada aresta desse ciclo recebe um valor unitário, enquanto os outros arcos recebem valores aleatórios. Instâncias das classes $G3$ e $G4$ são geradas aleatoriamente. No entanto, $G3$ possui instâncias esparsas (com $m = 4n$), enquanto que instâncias de $G4$ são densas (com $m = \frac{n^2}{4}$). Finalmente, as classes $G5$, $G6$ e $G7$ são instâncias de grade quadrada, longa e larga, respectivamente. Mais detalhes sobre estas instâncias podem ser obtidos em [Cherkassky (1996)]. Como as instâncias geradas pelo SPLIB contém apenas um objetivo, adicionamos três novos objetivos: além dos dois objetivos adicionados às instâncias de G_1, adicionamos também um valor igual ao inverso do valor do primeiro objetivo (objetivo-2), inserindo assim um objetivo com baixa correlação ao primeiro. Para cada classe foram criados grupos de três instâncias com dimensões diferentes, assim podemos analisar o comportamento do algoritmo em relação às dimensões da rede.

Nas instâncias de $G_2, .., G_7$ não há garantia de haver um caminho direcionado entre cada par de nós. No entanto, o gerador SPLIB informa um nó que garantidamente possui caminho direcionado a todos os demais nós. Desta forma, para estas instâncias consideramos este como nó inicial, enquanto que para as instâncias de G_1 consideramos $s = 1$.

A Tabela 2 apresenta o número médio de soluções eficientes do nó inicial a todos os outros nós do grafo. Esta tabela também informa as características das instâncias. Um "-" na tabela indica que o valor é desconhecido, pois o algoritmo extrapolou 3600 segundos de execução. As colunas $|\overline{S_{12}}|$, $|\overline{S_{13}}|$ e $|\overline{S_{14}}|$ apresentam o número médio de soluções não dominadas entre s e todos os demais nós do grafo quando considerando os dois objetivos indicados no índice de S. $|\overline{S}|$ representa o número de soluções considerando os quatro objetivos de cada instância.

Observa-se que mesmo para instâncias grandes, ou bastante densas como G_3 e G_4, temos um baixo número médio de soluções eficientes quando trata-se de apenas dois

objetivos. Exceto para instâncias em forma de grade que em comparação com as demais, apresentou um maior número de soluções eficientes. Aumentar os seus tamanhos não parece ter um acréscimo significante no número de soluções eficientes. Em [Raith (2009)] isso já foi observado e os autores não obtiveram sucesso em gerar redes randomicamente com um número grande de soluções eficientes sem que a rede tenha uma estrutura de grade. Ainda, o número de soluções eficientes é altamente dependente da correlação entre os objetivos. O conjunto $|S_{13}|$ que apresenta uma correlação muito baixa foi onde obtemos um número grande de soluções eficientes para a maior parte das instâncias, principalmente nas instâncias em forma de grade. Inversamente, objetivos correlacionados diminuem consideravelmente o número de soluções eficientes, como podemos perceber quando considerando apenas distância $\times$ *hop count* $|S_{14}|$. Quando isso acontece, ambos algoritmos têm um tempo de execução muito parecido, isso ocorre devido ao fato de que com menos soluções, menores são os conjunto de rótulos de cada nó e, consequentemente, as otimizações presentes no *merge* original acabam tendo um ganho muito baixo em comparação ao *merge* generalizado.

Nome	Classe	Nodos	Arcos	$\overline{\lvert S_{12}\rvert}$	$\overline{\lvert S_{13}\rvert}$	$\overline{\lvert S_{14}\rvert}$	$\overline{\lvert S\rvert}$
Chicago Regional	G_1	1202	4720	231.6	89.0	30.2	-
Chicago Principais	G_1	933	2950	8.1	10.6	4.3	83.1
Philadelphia	G_1	13389	40003	71.7	49.6	13.7	-
Sioux Falls	G_1	24	76	2.4	1.5	1.45	3.4
Acyc_1	G_2	8192	131072	15.0	6.8	11.3	45.3
Acyc_2	G_2	16384	262144	16.2	5.7	12.4	47.3
Acyc_3	G_2	32768	524288	16.4	7.3	12.5	51.3
Rand4_1	G_3	8192	32768	18.6	7.0	14.3	53.9
Rand4_2	G_3	16384	65536	20.7	7.2	16.7	63.9
Rand4_3	G_3	32768	131072	20.9	9.5	16.5	76.5
Rand1:4_1	G_4	256	32768	24.6	32.0	8.6	114.5
Rand1:4_2	G_4	512	65536	24.2	25.3	7.9	136.9
Rand1:4_3	G_4	1024	262144	27.3	38.8	8.4	202.5
Grid_sq_1	G_5	1025	3072	24.5	50.0	5.2	535.0
Grid_sq_2	G_5	4097	12288	80.6	176.9	9.3	-
Grid_sq_3	G_5	16385	49152	183.3	521.1	15.0	-
Grid_long_1	G_6	513	1536	81.1	147.3	7.0	-
Grid_long_2	G_6	1025	3072	246.1	439.0	10.9	-
Grid_long_3	G_6	2049	6144	688.1	1975.6	35.1	-
Grid_wide_1	G_7	513	1536	4.2	8.3	2.1	18.0
Grid_wide_2	G_7	1025	3072	4.6	8.4	2.1	17.3
Grid_wide_3	G_7	2049	6144	4.8	8.5	2.3	18.1

Tabela 2. Descrição do conjunto de instâncias de cada classe, com número de nodos, arcos, e a quantidade de soluções médias obtidas a partir nodo inicial a todos os outros nodos da rede, com relação ao subconjunto de objetivos utilizado.

Na Tabela 3 são mostrados os tempos de execução dos algoritmos CRB e CRM considerando conjuntos de dois objetivos, assim como o algoritmo CRM com todos os objetivos (duas últimas colunas). O número de soluções eficientes encontrado em cada execução é adicionado nesta tabela com o objetivo de tornar explícita a relação de tempo com o número de soluções eficientes.

Do ponto de vista de eficiência, considerando apenas os subconjuntos de dois objetivos, o algoritmo CRM obteve resultados mais próximos ao algoritmo CRB otimizado para biobjetivo nas redes da classe G_2 e G_3 com um tempo médio de 103,4 e 80,6 segundos, respectivamente, comparado com o algoritmo CRB que gastou 84,4 e 72,2 segundos, respectivamente. Portanto, o tempo do CRM piorou em apenas $11,7\%$ para as instâncias de G_2 e $22,5\%$ para as instâncias de G_3. Porém, quando aumenta o número de soluções

eficientes, o peso das otimizações originais começam a fazer uma grande diferença no tempo de execução para o CRM. Por exemplo, a instância Chicago Reginal, com 103 caminhos eficientes para os objetivos distância $\times$ aleatório, demorou 780% a mais do que o CRB. Em contraste, para a mesma instância, considerando apenas os objetivos distância $\times$ inverso-distância, o número de soluções eficientes cai para 43, e o CRM torna-se apenas 397% mais demorado. Contudo, nas redes de grades G_5 e G_6 o tempo aumentos em média 2800%, levando em consideração os valores que terminaram dentro de uma hora. Dessa forma, observamos que a quantidade de soluções eficientes é um dos fatores que mais influencia no tempo de execução do CRM. Esse resultado é esperado visto que o *merge* neste algoritmo é quadrático, enquanto que é linear em CRB. Ainda, tanto para CRB como para CRM, o tempo é bastante relacionado à classe de instâncias utilizada. Para alguns tipos de instâncias dificilmente conseguiremos obter um grande número de soluções eficientes, porém em outras, como as de grade, a própria estrutura do problema permite um número grande de caminhos eficientes.

No caso do CRM considerando todos os quatro objetivos, como era esperado o tempo e o número de soluções aumentaram consideravelmente. As instâncias das classes G_5 e G_6 gastam mais de 1h até para instâncias de menor tamanho, como a *Grid_sq_1*. Para as demais instâncias de grade o algoritmo CRM ultrapassou 3600 segundos de execução. [Guerriero (2001)] também verificaram os tempos de execução de algoritmos de correção de rótulos para instâncias em forma de grade utilizando até quatro objetivos. Mesmo utilizando instâncias relativamente pequenas em comparação às utilizadas nesse artigo, de 100 até 625 nós, obtiveram tempos de execução bastante elevados. Já para as redes de G_2 e G_3, o número de soluções aumentou aproximadamente 3 vezes em média em relação a dois objetivos, pois são grafos com densidade não muito alta. Já para instâncias da classe G_4 o número de soluções aumenta em média 7 vezes mais, devido a alta densidade da rede.

Analisamos também o tempo de execução gasto com os procedimentos Merge-B otimizado para biobjetivo e Merge-M generalizado para multiobjetivo. Lembrando que o tempo de execução do Merge-M é quadrático, enquanto que o Merge-B é linear. Podemos ver claramente que o procedimento *merge* é a parte mais custosa do algoritmo, Merge-B consumiu em média $66,6\%$ do tempo total de execução do programa; já o procedimento Merge-M consumiu $82,2\%$ do tempo total. Podemos notar também que o número de execuções do procedimento Merge-B é menor do que o Merge-M. Isso ocorre devido a verificação rápida de pré-dominância efetuada antes do procedimento *merge* descartando arestas mais custosas. Experimentos analisando a eficiência dessa verificação já foram constatados em [Skriver (2000)b] e são bastante relacionados aos tipos de instâncias. Nas intâncias com estruturas de grade o ganho com essa pré-verificação é quase nulo, menos de $0,1\%$. Contudo, em instâncias do tipo G_2 o maior ganho foi 39%. No Merge-M considerando todos os quatro objetivos, temos uma quantidade de execuções do *merge* bastante alta, devido ao grande aumento do tamanho do conjunto de rótulos não dominados dos nós da rede, os nós são marcados muitas vezes para verificação, aumentando a quantidade de chamadas ao *merge*.

	distância × aleatório			distância × inv. distância			distância × *hop count*				
Nome	$\|S_{12}\|$	CRB	CRM	$\|S_{13}\|$	CRB	CRM	$\|S_{14}\|$	CRB	CRM	$\|S\|$	CRM
Chicago Reg.	133	226,6	2006,2	43	62,0	310,8	28	23,3	42,8	-	>3600
Chicago Princ.	8	0,1	0,1	5	0,1	0,1	4	0,1	0,4	50	7.6
Philadelphia	14	82,9	236,2	10	36,8	88,6	6	6,5	10,0	-	>3600
Sioux Falls	2	0,1	0,1	1	0,1	0,1	2	0,1	0,1	2	0,1
Acyc_1	22	16,5	22,8	16	3,5	4,4	13	8,9	13,2	67	223,6
Acyc_2	24	53,6	70,5	5	8,1	10,7	12	25,2	38,6	46	603,1
Acyc_3	20	183,2	217,0	13	53,6	59,2	14	91,3	114,8	86	1599,6
Rand4_1	22	10,1	13,2	2	2,4	2,5	16	5,3	6,5	74	91,9
Rand4_2	21	43,5	51,4	6	10,8	11,0	17	23,5	27,5	75	326,9
Rand4_3	17	162,8	177,2	11	68,6	71,2	15	83,1	90,5	71	1205,2
Rand1:4_1	25	2,9	6,1	41	1,6	4,4	6	0,3	1,4	171	153,0
Rand1:4_2	31	6,1	12,6	31	2,5	5,6	9	0,5	2,4	179	524,7
Rand1:4_3	21	28,1	58,5	37	19,3	53,2	9	1,8	10,9	184	>3600
Grid_sq_1	30	0,3	0,8	128	0,6	2,6	10	0,2	0,4 1401	679,0	
Grid_sq_2	173	6,4	31,3	339	15,2	155,8	22	0,6	1,0	-	>3600
Grid_sq_3	395	149,1	1179,9	1290	494,8	>3600	22	11,0	15,0	-	>3600
Grid_long_1	166	0,6	2,7	375	1,6	15,1	14	0,1	0,1	-	>3600
Grid_long_2	597	6,4	74,9	1271	13,3	306,1	20	0,2	0,4	-	>3600
Grid_long_3	1932	110,2	3322,7	5682	346,2	>3600	67	2,7	6,3	-	>3600
Grid_wide_1	6	0,1	0,1	6	0,1	0,1	4	0,1	0,1	2	0,3
Grid_wide_2	5	0,1	0,1	19	0,1	0,1	3	0,1	0,1	1	0,5
Grid_wide_3	8	0,1	0,1	14	0,2	0,2	3	0,1	0,1	2	1,1

Tabela 3. Tempos de execução (em segundos) de CRB e CRM para o subconjunto de objetivos selecionados.

	objetivos: $\{1, 3\}$				objetivos: $\{1, 2, 3, 4\}$	
Nome	Merge-B	$\#Exe$	Merge-M	$\#Exe$	Merge-M	$\#Exe$
Chicago Regional	40.5	676047	287.2	684719	>3600	-
Chicago Principais	0.1	6291	0.1	7638	7.4	10237
Philadelphia	16.3	684208	67.3	701044	>3600	-
Sioux Falls	0.01	36	0.01	76	0.1	112
Acyc_1	0.8	205084	1.9	276602	207.6	2059852
Acyc_2	1.1	333881	3.2	539759	518.9	5409013
Acyc_3	4.4	981255	9.7	1257169	1217	10157186
Rand4_1	0.3	82852	0.5	91640	74.5	753429
Rand4_2	0.7	179529	1.2	197231	230.0	1752653
Rand4_3	2.2	446746	3.7	470531	718.8	3860864
Rand1:4_1	1.3	101067	4.1	109822	156.4	276539
Rand1:4_2	2.0	184109	5.1	191374	520.1	640244
Rand1:4_3	15.3	934302	49.5	969300	3648.2	2343245
Grid_sq_1	0.4	16212	2.5	16349	682.2	22561
Grid_sq_2	12.1	106635	151.7	106907	>3600	-
Grid_sq_3	376.4	835387	>3600	-	>3600	-
Grid_long_1	1.4	15508	14,7	15588	>3600	-
Grid_long_2	10.9	38092	302,7	38321	>3600	-
Grid_long_3	272.9	200253	>3600	-	>3600	-
Grid_wide_1	0.1	3409	0.1	3622	0.2	4393
Grid_wide_2	0.1	7123	0.1	7560	0.3	8974
Grid_wide_3	0.1	14140	0.1	14922	0.6	18287

Tabela 4. Merge-B e Merge-M são os tempos de execução gastos em cada procedimento em segundos, e $\#Exe$ é a quantidade de vezes que cada procedimento foi executada para o subconjunto de objetivos considerados.

5 Considerações Finais

Este artigo apresenta uma revisão dos métodos exatos para resolução de problemas BSP e MSP, e propõe uma generalização direta do algoritmo de correção de rótulos biobjetivo. Resultados computacionais foram executados com o objetivo de i) apresentar o número de soluções eficientes considerando sete classes de instâncias; ii) comparar os tempos computacionais considerando o algoritmo original (CRB) e o generalizado (CRM); iii) comparar resultados ao considerar objetivos com diferentes valores de correlação; iv) verificar o tempo total do algoritmo gasto com o *merge*. A partir dos experimentos concluiu-se que o número de soluções eficientes e o tempo de execução são bastante dependentes do tipo de instância. Ainda, os objetivos considerados também influenciam bastante os resultados: objetivos correlacionados tendem a produzir um menor número de soluções eficientes e, portanto, gastam menos tempo de execução.

Referências

Brumbaugh-Smith, J. and Shier, D. (1989). An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43(2):216–224.

Carlyle, M., W., and Kevin Wood, R. (2005). Near-shortest and k-shortest simple paths. *Netw.*, 46:98–109.

Cherkassky, B. V., Goldberg, A. V., and Radzik, T. (1996). Shortest paths algorithms: theory and experimental evaluation. *Mathematical Programming*, 73:129–174.

Cherkassky, G. and Radzik (1994). http://www.avglab.com/andrew/soft/splib.tar.

Climaco, N., Carlos, J., and Martins, E. Q. V. (1982). A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11(4):399–404.

Corley, H. W. and Moon, I. D. (1985). Shortest paths in networks with vector weights. *Journal of Optimization Theory and Applications*, 46:79–86. 10.1007/BF00938761.

Current, J. and Marsh, M. (1993). Multiobjective transportation network design and routing problems: Taxonomy and annotation. *European Journal of Operational Research*, 65(1):4–19.

Demetrescu, C., Goldberg, A., and Johnson, D. (2006). http://www.dis.uniroma1.it/ challenge9. Online. Acessado em 09/03/2008.

Gera, H. B. (2001). http://www.bgu.ac.il/ bargera/tntp/.

Goldberg, A. V., Kaplan, H., and Werneck, R. F. (2007). Better landmarks within reach. In *WEA*, pages 38–51.

Guerriero, F. and Musmanno, R. (2001). Label correcting methods to solve multicriteria shortest path problems. *J. Optim. Theory Appl.*, 111:589–613.

Hansen, P. (1980). Bicriterion path problems. In Fandel, G. and Gal, T., editors, *Multiple Criteria Decision Making: Theory and Applications, LNEMS 177*, pages 109–127. Springer-Verlag, Berlin.

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2008). Anytime search in dynamic graphs. *Artif. Intell.*, 172:1613–1643.

Martins, E. Q. V. (1984). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245.

Martins EQV, D. S. J. (1999). The labelling algorithm for the multiobjective shortest path problem. Technical report.

Mote, J., Murthy, I., and Olson, D. L. (1991). A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53(1):81 – 92.

Raith, A. and Ehrgott, M. (2009). A comparison of solution strategies for biobjective shortest path problems. *Comput. Oper. Res.*, 36:1299–1331.

Serafini, P. (1986). Some considerations about computational complexity for multiobjective combinatorial problems. In Jahn, J. and Krabs, W., editors, *Recent advances and historical development of vector optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems*, Berlin. Springer-Verlag.

Skriver, A. J. V. and Andersen, K. A. (2000a). A classification on bicriterion shortest path (bsp) algorithms. *Asia-Pacific Journal of Operational Research*, 17:199 – 212.

Skriver, A. J. V. and Andersen, K. A. (2000b). A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27(6):507 – 524.

Tung, C. T. and Chew, K. L. (1992). A multicriteria pareto-optimal path algorithm. *European Journal of Operational Research*, 62(2):203 – 209.

Vincke, P. (1974). Problèmes multicritères. *Cahiers du CERO*, 16.